Jonas Nyberg

Development of a Universal EtherCAT-Based Fieldbus Module

School of Electrical Engineering

Thesis submitted in partial fulfillment of the requirement for the degree of Master of Science in Technology

Espoo 27.5.2013

Thesis supervisor:

Professor Raimo Sepponen

Thesis instructor:

Ari Lindvall

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Jonas Nyberg

Title: Development of a Universal EtherCAT-Based Fieldbus Module

Date: 27.5.2013 Language: English Number of pages: 10+47

Department of Electronics

Professorship: Electronics and applications Code: S3007

Supervisor: Prof. Raimo Sepponen

Instructor: Ari Lindvall

A product development project was set up at the Finnish company SKS Control Oy in order to renew an older product portfolio consisting of a number of different devices used in electrically implemented motion control systems. These products include, for example, a range of programmable CPU units, display devices, HMIs, and I/O devices. This work focuses on the process of developing one of the new products, which is an EtherCAT-based fieldbus module. The purpose of this product is to replace a number of older products by integrating their functionality into one extensive modular-structured device. Along with this, the new module is intended to provide various new features that were hard or impossible to implement with the older products.

The part of the product development work which is presented here consists of three main parts. The first one presents some of the most relevant theoretical background behind the field-bus module along with a list of wanted features and properties. Together with the theory, this part discusses a way of satisfying these design constraints while using a module structure similar to the other products in the portfolio.

Second, the work covers the hardware design of the module by dividing it into smaller functional blocks and discussing them separately. Here the focus lies on the electronic design rather than the mechanical.

Finally, the work includes a section about the software work included in the product development. This part mainly focuses on the hardware description language VHDL and, to a smaller extent, on additional software, such as C, used in the product or together with it.

Keywords: EtherCAT, FPGA programming, VHDL, XML, C, I/O, PCB design, product design, field-bus

AALTO YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Jonas Nyberg

Työn nimi: Development of a Universal EtherCAT-Based Fieldbus Module

Päivämäärä: 27.5.2013 Kieli: Englanti Sivumäärä: 10+47

Elektroniikan laitos

Professuuri: Elektroniikka ja sovellukset Koodi: S3007

Valvoja: Prof. Raimo Sepponen

Ohjaaja: Ari Lindvall

Suomalaisessa yrityksessä SKS Control Oy:ssa aloitettiin tuotekehitysprojekti jonka tarkoitus oli uusia eräs vanha tuoteportfolio. Portfolio sisältää tuotteita jotka käytetään sähköiseen liikkeenohjauksen yhteydessä ja portfoliossa löytyy esimerkiksi erilaisia CPU-moduuleja, näyttöjä, HMI:ta ja I/O laitteita ym. Tämä kirjoitelma keskittyy yhteen uuteen tulevaan moduuliin, joka on EtherCAT pohjanen kenttäväylämoduuli. Tämän uuden tuotteen tarkoitus on korvata muutama vanha tuote ja samalla tuoda uusia ominaisuuksia jotka olisi ollut vaikeita tai mahdottomia toteuttaa vanhoilla tuotteilla.

Sitä osaa tuotekehitysprojektia joka esitetään tässä työssä sisältää kolmea eri osaa. Ensimmäinen osa käsittelee olennaisimmat alueet tuotteen teoreettisesta taustasta ja samalla esittelee tuotteen halutut spesifikaatiot ja ominaisuudet. Tämän yhteydessä esitetään myös erilaiset keinot toteuttaa nämä halutut ominaisuudet.

Työn toinen osa käsittelee tuotteen rautapuolen suunnittelua jakamalla kaikki toiminnalliset osat pienempiin osakokonaisuuksiin ja käsittelemällä niitä erikseen. Tässä osassa keskitytään enemmän elektroniikan suunnitteluun kun mekaaniseen.

Lopuksi työ käsittelee tuotteeseen liittyvää ohjelmointiosuutta joka pääosin käsittelee rautapuolen ohjelmointia VHDL kielellä ja pienemmissä määrissä muita tuotteeseen liittyviä ohjelmointikieliä kuten C.

Avainsanat: EtherCAT, FPGA ohjelmointi, VHDL, XML, C, I/O, PCB suunnittelu, tuotesuunnittelu, kenttäväylä

Preface

First of all, I want to thank my thesis instructor Ari Lindvall for granting me the opportunity to complete my Master's degree at SKS Control Oy in form of an interesting product development project.

Least but not last I would like to thank both Raimo and Ari for their time they took by reading through the work and correcting and commenting on it.

Sipoo, 27.5.2013

Jonas Nyberg

Contents

Symbols and abbreviations

ABS	Absolute
AoE	ADS over EtherCAT
ASIC	Application specific integrated circuit
A/D	Analog to digital
BGA	Ball-grid array
CMOS	Complementary metal oxide semiconductor
CoE	CANopen over EtherCAT
CPLD	Complex programmable logic device
CPU	Central processing unit
DC	Direct current / Distributed clocks
DCM	Digital clock manager
DPRAM	Dual port random access memory
DSP	Digital signal processing
D/A	Digital to analog
EEPROM	Electronically erasable programmable read-only memory
EMC	Electromagnetic compatibility
ENI	EtherCAT network information
EoE	Ethernet over EtherCAT
EOF	End of frame (EtherCAT frame)
ESC	EtherCAT slave controller
ESI	EtherCAT slave information
ESM	EtherCAT state machine
EtherCAT	Ethernet for control and automation technology
ETG	EtherCAT technology group
FCS	Frame check sequence
FET	Field-effect transistor
FLASH	Non-volatile computer storage memory
FMMU	Fieldbus memory management unit
FPGA	Field programmable gate array
FR4	Grade designation assigned to PCBs' core material
GPIO	General purpose Input / Output
I^2C	Inter-integrated circuit two-wire interface
IC	Integrated circuit

ID	Identification
INC	Increment
IEC	International electro-technical commission
IEEE	Institute of electrical and electronics engineers
IP	Internet protocol
IP	Intellectual property
ISO	International organization for standardization
I/O	Input / Output
JTAG	Joint test action group
LCD	Liquid crystal display
LED	Light emitting diode
LUT	Lookup table
LVDS	Low voltage differential signaling
MAC	Media access control
MMU	Memory management unit
MUX	Multiplexer
NIC	Network interface controller
PAL	Programmable array logic
PC	Personal computer
PCB	Printed circuit board
PDI	Process data interface
PDO	Process data object
PE	Protection earth
PHY	Physical layer
PLD	Programmable logic device
PLL	Phase-locked loop
PTC	Positive temperature coefficient
QFN	Quad-flat no-leads
RJ45	A common connector type used for Ethernet
RTOS	Real-time operating system
R/W	Read / Write
SEMI	Semiconductor equipment and materials international
SFR	Special function register
SM	Sync manager
SMD	Surface-mount device

SMPS	Switched-mode power supply
SoC	System on chip
SoE	Servo profile over EtherCAT
SOF	Start of frame (EtherCAT frame)
SPI	Serial peripheral interface bus
SRAM	Static random access memory
SSI	Synchronous serial interface
TTL	Transistor-transistor logic
TVS	Transient voltage suppression
UDP	User datagram protocol
VHDL	VHSIC hardware description language
VHSIC	Very high speed integrated circuit
VLAN	Virtual local area network
VoE	Vendor specific profile over EtherCAT
WD	Watchdog
XML	Extensible markup language

1 Introduction and objectives

As the demand for faster performing and more complex automation systems arises, the need for new innovations and solutions is brought into light and has to be answered by the automation technology companies. One such cost-effective solution that provides the customer with an efficient and flexible system is the real-time EtherCAT® fieldbus system. By using the same underlying commodity technology in the EtherCAT® system as with standard Ethernet, which is mainly driven by the office sector, the overall system cost reduces significantly. The office Ethernet technology does not only bring cost-reduction, but also gives rise to another one of the main advantages with EtherCAT®, namely that it makes allowance for internet access (e.g. webserver) within the same system. The internet access can quite possibly reduce the number of interfaces in the automation system as nowadays system access over the internet is widely implemented. However, comparing EtherCAT® with the office Ethernet shows clearly that EtherCAT® is better suited for automation systems than the standard office Ethernet technology. One reason for this is the lack of real-time operation with the office Ethernet. Another reason is the small utilization of the Ethernet packages and slower speed due to the fact that the packages have to be received, copied and interpreted before forwarding. This is not the case with EtherCAT® as will be seen later in the thesis.

EtherCAT® differs from standard Ethernet in many ways by the most prominent one being the determinacy i.e. real-time characteristics. The EtherCAT® real-time capability, using time-stamps based on a distributed clock system, allows for faster system response and easier synchronization of modules in larger systems, which is a much desired feature due to the fact that some grade of synchronization is involved in a large number of automation systems. The EtherCAT® fieldbus competes with other similar real-time solutions on the market, with the most visible ones being the Siemens Profinet® industrial Ethernet and Powerlink® systems, all with their own pros and cons.

The root of a simple automation system using the EtherCAT® fieldbus technology consists of one master controller and one or more EtherCAT® slave(s). The master can but does not have to be a normal PC, with suitable software and hardware, and is usually connected via standard Ethernet cables to the slaves, without the need of any hubs or switches in between. The EtherCAT® master controller is nothing more than a software program that usually uses the same network controllers and hardware as the slaves in the Ethernet physical layer. The master creates and sends out the EtherCAT® frames or packages to be forwarded by the slaves. The slave's role is to forward the EtherCAT® frames on the fly, while simultaneously extract and insert data into the frames which has been specifically addressed for each slave. The master also controls and executes the application specific automation program by using the slaves as the electrical interfaces to the other devices in the system. The program's complexity can for example range from a simple I/O reading task to a much more complex motion control task involving sophisticated synchronous motion of several axes.

A product development project was set up at the Finnish company SKS Control Oy, with the main objective being to develop a competitive EtherCAT®-based fieldbus module as a part of a product portfolio modernizing project. The new products to be developed in the modernizing process are meant to replace an old much wider product portfolio by integrating more functionality into fever products and make use of the other

benefits that EtherCAT® gives rise to compared to the old system. The two most central products of the portfolio are an EtherCAT® master CPU module and an EtherCAT® slave I/O module, of which the latter one is the subject of this thesis. Although the project covers the whole product development process from idea to the complete sales-ready product, the scope of this thesis doesn't cover all parts in the process as will be explained later.

My personal part of this product development project involving this particular EtherCAT® slave module is quite comprehensive. It involves all the hardware prototyping and design, all the software design and the major part of the documentation. However, areas such as sales & marketing, some of the ideation around the product features and the final production management are outside my field of work.

The objective of this thesis is to provide the reader with comprehensive information about the work that has been done and knowledge that can be used for similar future development, such as new spin-off products or updated versions of the original one. The thesis simultaneously stands as a technical documentation, leaving out some of the deeper knowledge behind the implementations due to the fact that they are trade secrets. This product documentation stands alongside with other documentation such as the user's manual and technical datasheets etc.

1.1 Product overview

As shortly stated before, the product that is being discussed in this thesis is an EtherCAT slave fieldbus module. The module is operated by a master device connected to it using standard Ethernet NICs that are compatible with the EtherCAT protocol. The specifications for the EtherCAT protocol allow one master to be connected to one or more slaves of different or the same type. Because the protocol is standardized and all vendors have to develop their products according to it, the master and the slave modules don't have to be from the same vendor or product family to work together. The protocol also supports different network topologies, such as a line-, tree-, star- and ring-topology or an arbitrary combination of them all. These different possibilities give the system architect more freedom while simultaneously adding more qualities to the system, such as the possibility for redundancy. Communication is not only restricted from master to the slaves and vice versa but also allows for a fast slave to slave communication.

The electronics of the slave module itself is encapsulated inside a plastic box consisting of one motherboard, with electrical connectors, upon which a number of PCBs with different functionality can be stacked, to fulfill the required specifications of the system. This allow making one more basic and cheaper version of the product for simpler projects, containing only the motherboard without add-ons, but with some basic functionality integrated, such as digital I/O. On the contrary, if a more complex system is needed, there is a possibility to use a number of different add-on PCBs, with different functions and electrical connections, by stacking them onto the motherboard. The add-on PCBs, which are referred to as option modules thorough the thesis, provide the module with more complex functions such as A/D-, D/A-converters, ABS- and INC-encoder interfaces, additional I/O with special properties etc. A more thorough description of the possibilities this modularity gives rise to is given later on.

1.2 Product development goals and comparison to similar products

The main goal of the product development is to manage the many needs of today's automation systems with one extensive modifiable fieldbus module. Therefore pressure is set to plan and design the module for maximum flexibility and adaptability. This adaptability will not only reduce the costs in some applications, but will also allow the module to be modified to meet the customers' specific needs in special cases. These special cases can be situations when no off-the-shelf module is available to satisfy some desired arbitrary function(s) and the alternative solution would be somehow cumbersome. One example of this could be a simple thing as measuring the frequency or pulse width to a high precision of a digital input channel. Another example could be applying a digital filter of variable length to digital inputs. These are examples of things not easily implemented by using simple off-the-shelf modules, but can be implemented into the module with a small effort using the hardware description language VHDL together with a programmable FPGA, as will be seen later.

Another goal is to keep the module and the building-blocks it consists of rather simple in construct and reduce the component amount as much as possible. This will help to manage the production costs and increase the reliability, while still providing a flexible and easy solution. Implementing an FGPA chip in the design which is programmed using the hardware description language VHDL will help to reduce the component count significantly. This is due to the fact that most of the complex hardware functions can be implemented inside one FPGA chip, simply by programming it.

A third goal related to the electrical design is to keep the motherboard as open as possible for future improvements. Specifically to keep the add-on PCB-board connectivity as flexible as possible to allow for more and easy improvement of the module properties by using future add-on boards of different kinds and complexity. The FPGA comes in handy here once again because new add-on boards can be taken into use by simply modifying the VHDL software and leaving the electrical connections the same as before.

Conventional EtherCAT systems sold by larger vendors, such as Beckhoff and Wago, all build their systems using one EtherCAT coupler, which is connected to a master, together with off-the-shelf EtherCAT slave modules stacked onto the coupler. This gives a similar modular construction as the one described in the chapter before and the system is built similarly by stacking the needed slave modules together. This method lacks the ability to arbitrarily modify each module for specific special needs, which is one downside with off-the-shelf modules.

The last but not least advantage with developing a module of one own is the in-depth knowledge and understanding of the product it provides. This will naturally give aid to the technical support and therefore also give more value to the customers.

1.3 The scope of the thesis

The scope of this thesis is mainly restricted to the hardware and software development of the EtherCAT-based fieldbus module, together with some basic theory of operation. Nevertheless, a short introduction is given regarding its use and role as a part of a bigger automation system including some discussion about communication and interfaces to other devices. The hardware part of the thesis focuses more on the electronic than the mechanical product design and gives the reader in-depth knowledge about the electronic building-blocks used and how they work together. In the same manner, the software part of the thesis discusses the software blocks which builds up a functioning product with the desired features and are programmed into the FPGA. It is assumed that the reader is familiar with the hardware description language VHDL. The thesis software part also slightly touches the subject of XML coding and its use in an EtherCAT system.

The thesis does not go in-depth discussing the product development process as a whole but rather focuses on the decisions already made and discusses them more closely. Although the product all together is built up using a varying number of PCB-boards connected to the motherboard, only the development of the motherboard itself is discussed closer. Also nothing is said about production and production related testing and verifying.

1.4 Thesis organization

The thesis is organized into five main parts, starting off with an introduction and overview of the work that has been accomplished. Along with this is a short description of the product and the system it is used with.

The second part discusses the architecture or build-up of the product along with some theory behind the features and technologies used. This chapter also includes a short discussion about the communication interfaces of the product and the role of the product as a part of a bigger automation system.

The third part is the hardware part of the thesis, which begins with an overview and layout of the product and then proceeds to discuss a short examination of the mechanical design and a more comprehensive one about the electronic schematic design, extended with sections considering the PCB design and component selection. Finally, testing and evaluation is discussed in the last subsection.

The fourth part examines the software design involved in the product development. First, there is a more in-depth discussion regarding the hardware description language VHDL. Second, there is a shorter discussion about the XML markup language used when interfacing the product to a master device. Last in this section is a discussion about testing and validating the software.

The fifth and last part of the thesis concerns testing and confirmation of function for the complete product and finally sums up everything analyzed and discussed throughout the thesis, together with some cogitation of future improvements.

2 Theory of the EtherCAT protocol and module operation

This chapter starts with a presentation of the EtherCAT protocol and the theory behind its functional principle along with the standard and organization behind it. After this, the fieldbus module is presented in greater detail and the function of the module as a part of an automation system together with its interfaces, is discussed. The last part will discuss the FPGA used in the product design along with the possibilities it allows for and why it is important for the module.

2.1 EtherCAT protocol

The idea of EtherCAT got its beginning in the millennium shift at the German-based company named Beckhoff and was presented to the world for the first time at the Hannover fair in 2003. The first EtherCAT fieldbuses were introduced in the same year, consisting of I/O terminals, encoder readers and drives [p. 18-21, 1]. These fieldbuses were at the time already used in pilot projects, in which conventional fieldbuses could not be used. One example of this is the Schuler AG press controllers, in which EtherCAT was used for communication between peripheral devices and a PC-based control system [p.22-25, 2].

2.1.1 The EtherCAT standard and ETG (EtherCAT technology group)

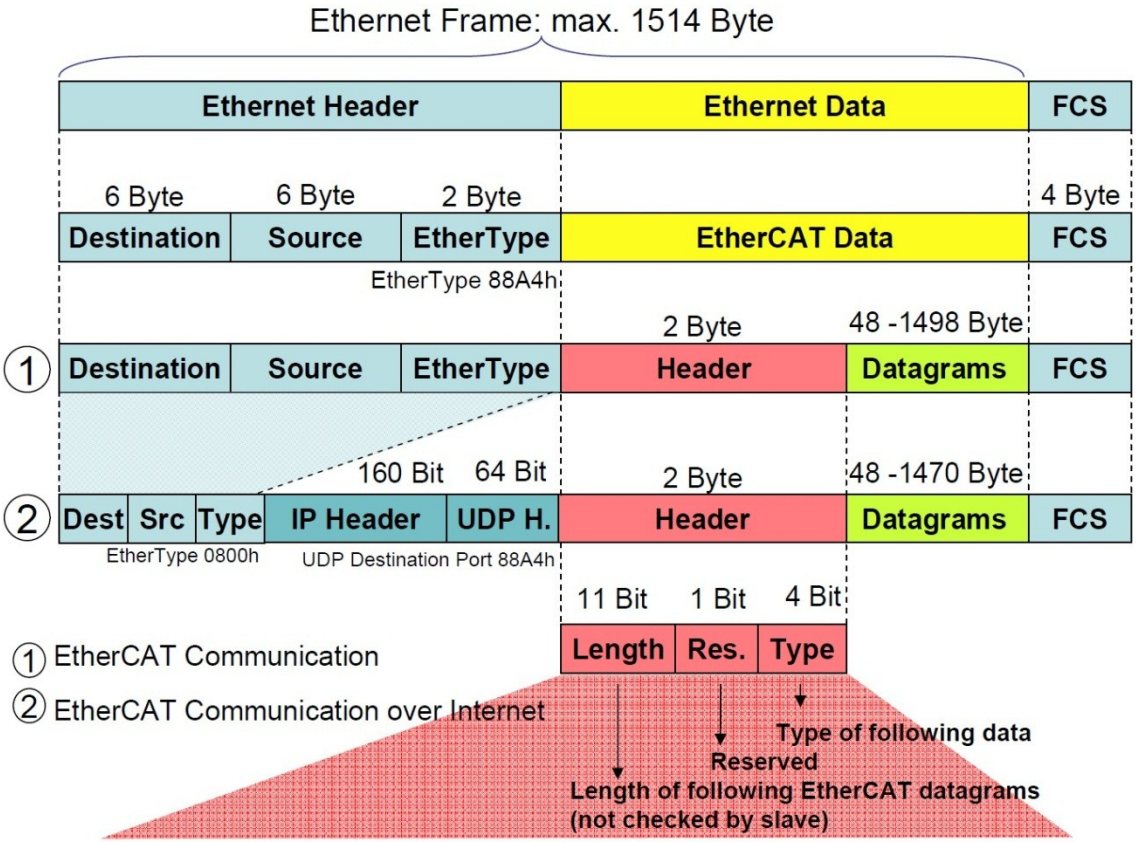
EtherCAT is an open source protocol and is an IEC, ISO and SEMI standard. To begin with, EtherCAT is part of IEC standards such as the IEC 61158, IEC 61784 and IEC 61800. The IEC 61158 and IEC 61784 are international fieldbus standards and one part of the IEC 61800 includes drive profile standards in which also the EtherCAT drive profile is included. Secondly, EtherCAT is also a part of ISO 15745, which deals with device descriptions. Finally, The SEMI organization has added the EtherCAT SEMI E54.20 to their standards. [p.7, 3]

EtherCAT is maintained and managed by the EtherCAT technology group, which today consists of about 2000 member companies worldwide. One of the fundamental ideas of the EtherCAT technology group is to encourage the members to influence the future enhancements of the open standard by attending technical working classes and other meetings. This gives each member company direct possibility to influence the development of EtherCAT and indirect possibility to represent one's interest to the national standardization companies, such as IEC and ISO, via ETG. [4]

2.1.2 Operating principle and frame processing

The data packets i.e. the Ethernet frames used by EtherCAT exert the same IEEE 802.3 standard as the frames used in the well familiar home or office Ethernet. First off, this allows for the use of standard network controllers and hardware on the master side. Second, EtherCAT has its own reserved EtherType, which allows for other protocols such as IP to be used parallel with the EtherCAT frames on the same bus. Alternatively EtherCAT frames can be encapsulated in UDP/IP frames or in VLAN frames. The real-time properties are not compromised even when using other protocols in parallel with the EtherCAT protocol in the same bus. On the next page, in picture 1, is an overview

of an Ethernet frame containing EtherCAT data raw and inside a UDP/IP internet frame. The first blue part from the left is the package header, which for a basic EtherCAT frame contains the destination, source and EtherType. The second (Yellow or red/green) part contains the EtherCAT data which can be further split into an EtherCAT header and one or more EtherCAT datagrams. The EtherCAT header contains the protocol type and info about the length of the EtherCAT datagrams. After the header, there are one or more datagrams which contain the raw data bits (EtherCAT configuration data and read/write process data) which are being transceived. The last part of the frame is the FCS or frame check sequence that is used for error detection.



Picture 1: Build-up of an Ethernet frame containing EtherCAT data.

No processor power is needed to process the frames as they are processed on the fly in hardware by the ESCs. Processing on the fly means that data are read and written as the bits are passing by the ESC and directly forwarded to the next closed port. This method gives the system a very low forwarding delay because the frames are not first stored, evaluated and then passed on, as is the case with the office Ethernet. [p.I-4 – I-5, 5]

2.1.3 The Physical layer, PHY

The EtherCAT physical layer consists of the components and connectors that make up the interface that transmit the EtherCAT frames from one device to another. The EtherCAT protocol supports two types of physical layers. The first one is the Ethernet interface mentioned earlier that uses a standard EtherCAT compatible MAC which have to support a 100Mbit/s full duplex link, according to the requirements for EtherCAT. The typical connectors used with the Ethernet approach are the same as in the normal office sector Ethernet, namely RJ45 connectors. The cable length with this method can

be up to 100m for copper cables and up to 2km for optical fiber. The second physical layer type is the E-bus, which is a LVDS bus intended to be used as a backplane module-to-module bus but can occasionally be used for communication up to 10m distances. The E-bus is designed to reduce components and costs while achieving the same 100Mbit/s speed as with the Ethernet link. More technical information regarding the E-bus is given in reference [p. I-34 – I-37, 5]. The physical layer can be changed at any time anywhere in the system and the Ethernet protocol approach allows for hot-plug-and-play compatibility. The only requirement for the transport medium is that is has to be full-duplex, leaving out solutions using half-duplex communication, such as radio transmissions. [p. 3, 6]

2.1.4 FMMU and SyncManagers

The data utilization with the EtherCAT protocol is usually very high and can come up to over 97% when an Ethernet frame is fully utilized. The high utilization grades come from the fact that several slaves in the system can be addressed in both send and receive direction within one EtherCAT frame by using logical addressing. This can be compared to a system the other way around where each slave has its own addressed Ethernet packet [p. 8-9, 3]. The logical addressing is made possible by the EtherCAT FMMU function. The addressing works by mapping each slave's data bits or bytes to a respective memory area within the FMMU's logical address space that spans 4GBytes in total. The FMMU function can be compared to a CPU's MMU and its function is to convert a logical memory address to a physical one via an internal table, within each slave. Accordingly, each slave uses one or more hardware configured FMMU's located in their respective ESCs to fulfill this function. The system can be regarded as a large distributed memory which can be written and read without restriction. The maximum data size of an Ethernet frame is 1500 byte, as can be seen in picture 1, which means that the whole memory area of the 4GByte address space is fragmented over several frames. [p. 2-3, 6]

To ensure that the data transfer between a master and a slave's local application is consistent and secure, a so called SyncManager or better described as a memory manager, is used. The SyncManagers are configured by the master and initialize a buffer for exchanging the data between master and slave. This buffer function results in that no polling of the memory is needed, to know when the other side (master or slave) has finished accessing the memory. The SyncManagers can be configured in two ways, the first one is a buffered mode and the second one is a mailbox mode. The buffered mode is typically used for cyclic process data and allows both the producer and consumer to access the buffer simultaneously. The buffer works as a 3-buffer with one reserved buffer for the consumer and one for the producer while the third one keeps the latest consistently written data by the producer. This function ensures that there is always the latest data available for the consumer. In contrast, the mailbox mode alternates the access to the buffer in a way that the consumer or producer has to finish its access first before the other side has access to the buffer. This works like a handshake mechanism to ensure that all data reaches the consumer from the provider and no data is lost. The mailbox mode is usually used for application layer protocols such as EoE, SoE and CoE, to name a few examples. [p. I-40 – I-44, 5]

2.1.5 Distributed clocks

One of the bigger advantages of EtherCAT compared to conventional Ethernet is the possibility to use the so called distributed clock (DC) feature which makes it possible for all EtherCAT devices to share the same EtherCAT system time. This does not only allow all the devices to be synchronized to each other, but also allow the local applications to be synchronized with each other as well. The clock synchronization works by defining one (usually the first ESC with DC compatibility) slave for holding the system reference time and then synchronizing all the other clocks (including the master's) to it. All the differences between the local clocks and the reference clock, such as drifting, offset and propagation delays can be accurately calculated, measured and compensated for with sophisticated compensation methods. For more information about how the system clock compensation is calculated the reader are encouraged to consult reference [I-45 – I-65, 5]. The ESCs can be configured to generate sync- and latch-signals synchronized to the EtherCAT system time for synchronous output signals and precise time-stamping for input signals.

While enabling synchronized clocks in the system, the time jitter between devices can achieve values of well below 1uS which can come in handy, for example in applications where several servo axes carry out synchronous coordination of movements. Naturally, this well-defined time reference is well-suited for motion control, where velocities are often calculated from sequentially measured positions. The distributed clocks can also be used to provide for accurate information about various local data acquisition and usually give better reaction times in the system. [p. 12-13, 3]

2.1.6 EtherCAT master and slave implementation

A master controller in an EtherCAT system does not need any special hardware to work and can thereby be fully implemented in software using standard on-board Ethernet MACs (in PC) or an additional passive NIC card as the physical interface to the slaves. The master functionality does not put much stress on the processor and can be implemented by using open-source projects or alternatively bought as commercial software. The master stack can be implemented on many RTOSs, not only including the well familiar Windows, Linux and OS-9 but also smaller ones such as CodeSYS RTOS. The burden on the host processor is eased due to the fact that the Ethernet frame is already sorted and all the mapping (the slave FMMU functionality) happens in the slaves i.e. the slaves insert their data at respective places in the passing frame. The master software stack is also applicable for embedded systems consisting of a broad range of microcontrollers or CPUs running different operating systems or no operating system at all. Even though embedded systems often have a limited amount of memory and possibly no hard disk, a freely sizeable and scalable version of the master stack can be implemented serving the need at hand. A freely scalable and sizeable master in this case means a master for which some of the features and/or protocols can be omitted if necessary, in order to reduce the size and therefore better work together with the memory and CPU used in the specific system. [7]

The EtherCAT protocol can support up to 65535 nodes or slaves in one network. Each slave has its own ESC, as described before, implemented either in a single ASIC chip or coded into an FPGA. An EEPROM is used together with the ASIC but not always with the FPGA (can be emulated with FPGAs), which holds the configuration and general information about the slave in question. At the system initialization, the master reads

this information from each slave and can gather information such as product ID, vendor, general PDI-configuration and distributed clocks settings etc. The PDI is the communication interface between the ESC and the local application. There are a number of different types of PDIs ranging from more simple I/O-wires to more complex 8/16-bit microcontroller interfaces and 32-bit parallel busses, to name a few. The more complex PDIs naturally need a microcontroller or CPU on the application side, while the simplest ones can be implemented without any. The PDI gives the application access to the ESC DPRAM (0-64kB), which is the internal ESC memory used for exchanging data with the master. The DPRAM is handled by the SyncManagers and further on mapped onto the Ethernet frames by the FMMU, as explained before. Some of the DPRAM is reserved for configuration and status data, but up to 60kB of RAM for each slave can be used by the local application for process data exchange with the master or other slaves. The memory size available is naturally dependent on the type of ESC used or its configuration, when talking about FPGAs. [p. 20-21, 3]

Furthermore, to coordinate the master and slave applications at start up and under operation, a so called EtherCAT state machine (ESM) is implemented in the ESCs. The function of the ESM is to initialize a controlled startup of the slave and to inform the master about possible problems or errors within the application. The master is the one that requests state changes of the slave and the slave answers accordingly by changing its state or jumping to a defined state in situations when an error or other unsuspected situation has occurred. In the different states of the slave, different types of communication and amounts of data exchange between the application and the master is allowed. For example, in a safe state, the slave is not allowed to change its outputs but is however allowed to read the inputs. This is an example how to avoid dangerous situations when some error or other problem has occurred. [I-66 – I-70, 5]

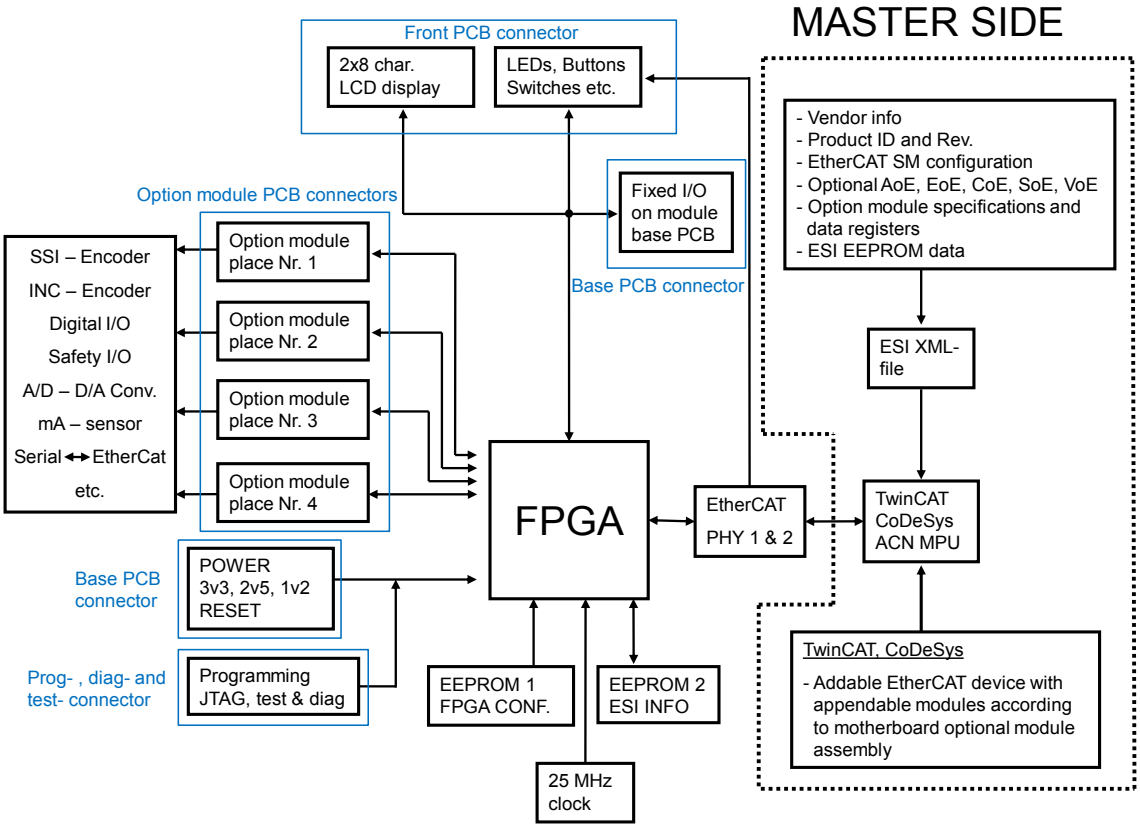
In summary, the simplest slave possible, which is a digital I/O, can be implemented by using only one EtherCAT ASIC or FPGA, one EEPROM and one Ethernet PHY together with their connectors and components. On one hand, this gives a very cost effective solution for simple devices and on the other hand even the simplest slaves share the same underlying technology as for more complex and costlier ones.

2.2 Module architecture and design

Now that the features of EtherCAT and the techniques behind EtherCAT slaves and masters have been covered, it is time to look closer at the slave that is being discussed in this thesis. As stated before, the slave module consists of one simple skeleton motherboard with the required minimum amount of components for a fully working slave. Additionally to this, there are a number of connectors on the motherboard used for interfacing other devices. First off, there are four option module connectors for add-on PCBs, as described before. Second, there is one connector for a base PCB containing the power input and the physical connectors for the always present digital I/O. Third, there is one additional connector for a PCB front panel containing an LCD display together with some buttons and LEDs. Finally, one connector is reserved for diagnostic, programming and testing purposes. The minimum component requirements consist of a programmable FPGA and two EEPROMs together with a power supply and other auxiliary components and connectors. The FPGA itself contains the ESC, which is configured to use the microcontroller PDI interface. The EEPROMs are needed for the ESI information and for holding the FPGA configuration file. No microcontroller is actually used as the interface is operated using a VHDL programmed module inside the

FPGA. All this, except for the base PCB, is contained inside a plastic box, giving the module non-changing physical dimensions independent of the number of option boards connected and is therefore also independent of the overall complexity of the module.

Below, in picture 2, an overall principal layout of the module's motherboard and its surroundings is presented. In the picture, the area inside the dotted line is the master side of the system, showing the EtherCAT master software alternatives and some elemental parts and configurations that go with them. The master part is shown here only for reference and will be discussed more in-depth in chapter 4.2.2 which concerns XMLs. Earlier in this chapter the non-dotted area were discussed, which consists of the FPGA in the middle together with all its connectors and possible add-on devices connected to the option module slots. The physical presence and layout of the module is presented later in chapter 3.1.



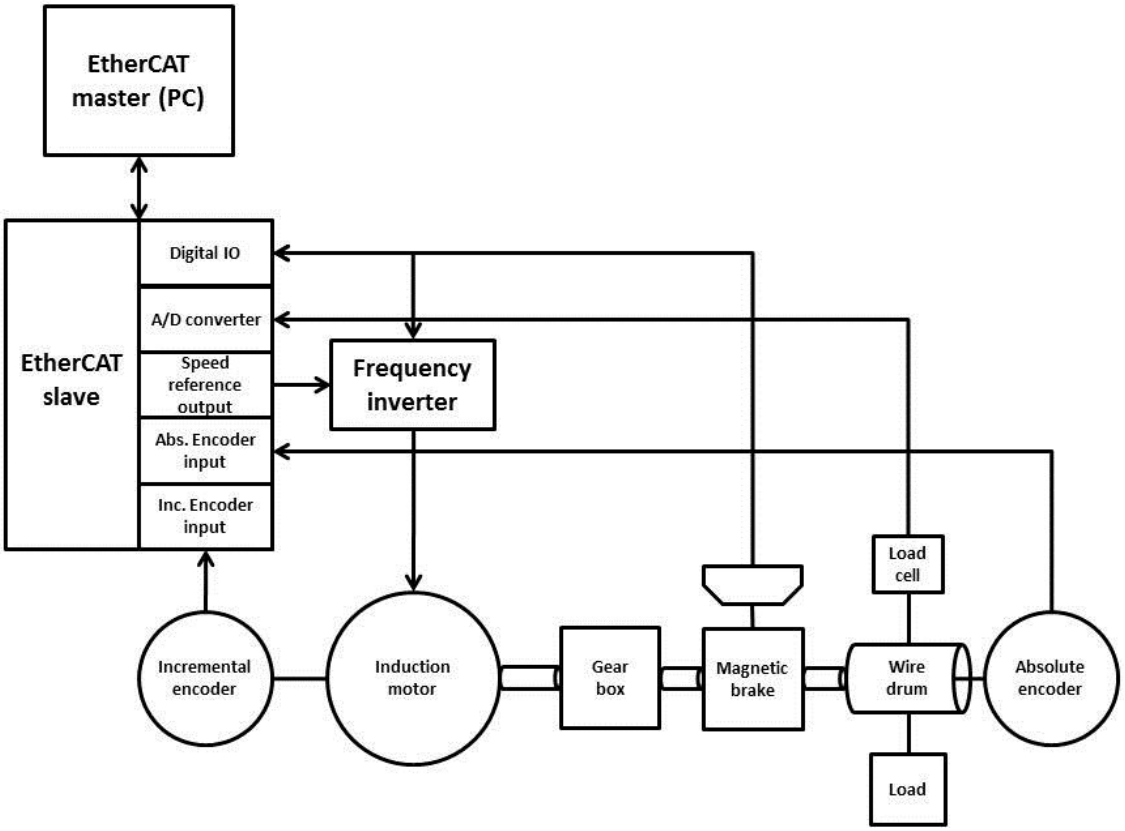
Picture 2: Module connections and functional part layout.

2.3 Module as a part of a larger automation system

As known from before, the EtherCAT slave module discussed in this thesis is usually a smaller part of a bigger automation system, possibly consisting of several EtherCAT slaves, servo- / motor-drives, motors, servos and hydraulics etc. The EtherCAT fieldbus system is suitable for a wide range of applications where the strengths such as easy implementation, flexibility, speed and cost-effectiveness come into play. Some typical applications in which EtherCAT fieldbus systems are used and well suited for often relate to some sort of machine controls e.g. metal forming, packaging, robotics and other complex motion control, assembly systems, printing machines and theater scenery control.

The EtherCAT slave module, which this thesis focuses on, handles the electrical connections and acts as the interface between the master and all possible devices to be connected to the EtherCAT fieldbus. These devices can range from simple relays and contactors directly connected to the digital I/O, to more complex analog input / output modules or encoder inputs. The analog inputs are often used to connect different types of sensors that use voltage or current outputs. The analog outputs however can be used in a motor control loop as the analog speed reference output to a servo drive. Different types of encoders are often used as position feedback from electrical motors in closed loop motor control systems. These examples do not include all the types of devices that can be connected or used, but rather the most typical ones seen in practical applications. A more comprehensive description of the module interfaces and connectable devices is given later in chapter 2.4.

A practical example of an automation system will be given next, including the slave module as an essential part providing for the major part of the functions and interfaces between the devices. The system is shown on the next page in picture 3 and is a hoist operated by an electrical motor which is controlled by a speed reference signal originating from the EtherCAT slave and connected to a frequency inverter. The motor control loop and the application interface of the hoist system itself is operated by software inside for example a PC, using suitable EtherCAT master software for communication over the fieldbus. The system uses one master and one slave that handle all the communication between all the devices. Two encoders, one incremental and one absolute are used for the position feedback to the EtherCAT master. The incremental encoder, used in the motor control loop for position feedback, is directly connected to the motor, while the absolute encoder is connected to the wire drum after the gearbox and brake. This gives the system a fail-safe mechanism in case of a gearbox breakdown, which cannot be seen by the control system using only the incremental encoder. A magnetic brake is connected to the shaft after the gearbox and is operated with the slave's digital I/O. The brake status i.e. is the brake open or close and possible failures can also be read with the digital inputs. The digital I/O is also connected to the inverter for enabling and disabling the drive at need. Further on, An A/D-converter inside the slave is connected to a load cell on the wire mechanism used for measuring the load and forwarding the value to the PC using the EtherCAT fieldbus.



Picture 3: Example system including the slave module

2.4 Interfaces to other systems and devices

Devices such as absolute or incremental encoders, A/D-D/A-converters and additional I/O are in first hand connected to the EtherCAT slave via the four option module connectors situated on the motherboard. The connectors provide for power and data communication between the option modules and the motherboard. Also, the option modules themselves provide for the same (data and possibly power) to the devices connected them. The digital I/O is connected to the base PCB and is routed to the motherboard via the motherboard's bottom connector. The connectors' physical placement on the product is shown in picture 5 in chapter 3.1.1.

2.4.1 Encoders

One of the device types that can be connected to the module via an option module is position encoders. The module will at first only accept rotary incremental and SSI-absolute encoders leaving out the option for Sin/Cos encoders which can be added in the future if needed. The INC- and SSI-ABS-encoders usually have a six- versus four-wire digital TTL interfaces which can be easily connected to the FPGA by using auxiliary driver and receiver buffers. The position is read, internally stored and sent via EtherCAT to the EtherCAT master of the system. These types of encoders are widely used in different industrial automation systems where the angular position or motion of an axle or shaft is needed.

In contrast to traditional EtherCAT encoder interfaces, the FPGA can be used to calculate additional information such as speed, acceleration etc. from the incremental encoders. This is done inside the FPGA by using logic to calculate the time difference

between the pulses and forwarding this information over EtherCAT to the master or using it internally. When this additional information is pre-processed in the FPGA, it does not only ease the burden on the master CPU but also provides very accurate speed and acceleration values usually not achievable with conventional methods. Another advantage that comes with the FPGA is the possibility to oversample and multiply the pulse count of the incremental encoders using interpolation between pulses. This can for example come in handy in cases where an expensive high resolution incremental encoder is needed but instead a cheaper one can be used together with the pulse multiplication inside the FPGA.

2.4.2 D/A and A/D converters

Modules converting analog signals (current or voltage) to digital and vice versa can also be interfaced to the FPGA via the module slots. The A/D-type of converter is often used in industrial automation together with different kinds of sensors providing voltage or current outputs. These sensors can be for example load cells, such as in the example in chapter 2.3 or temperature sensors or pressure sensors, to name a few. In practice, any type of sensor providing voltage or current output could be connected. Furthermore, the A/D option module could be multiplexed providing a number of analog inputs on the same physical module. The FPGA together with its internal RAM-memory could be used if needed to oversample the analog inputs in the means of taking more than one sample during one EtherCAT update cycle and sending them in bulk whenever asked for by the EtherCAT master.

The other type of module, the D/A-converter is in most cases used as a ±10V speed reference output module to inverters not accepting digital speed references. These inverters are usually cheaper and therefore preferred in some cases, compared to the ones that have a digital reference which could be sent over a fieldbus, such as EtherCAT. This type of module is also included in the example in chapter 2.3

2.4.3 Digital I/O

The digital I/Os of the slave module are situated on the bottom PCB and are connected to the FPGA through intelligent buffer ICs. The buffers are intelligent in the manner of automatic overload-, current limit-, short circuit-, over temperature- and over voltage protection. These buffers also have a diagnostic feedback output providing information to the user about different conditions such as overload, open-load, over temperature and short circuit. The I/Os are 24V high which is the typical voltage used in industrial automation.

The FPGA can also add some additional features to the digital I/O. The logic can for example be programmed to calculate input pulse widths or frequencies, which are of greater precision compared to the situation where the master CPU does the same operation. Naturally, this pre-processing of data also relieves stress from the master CPU. In addition to this feature, another example could be programmable digital filters which could be programmed into the FPGA and used with the digital inputs for damping glitches.

2.4.4 Additional and future modules

The option module types are not restricted to the modules described in the last three chapters and in practice any module with a digital interface to the FPGA can be added.

However, the module types described before are the only ones that are developed before the product is released on the market. Depending on future needs, modules such as serial-to-EtherCAT converters, CPU & memory modules, Safety I/O s and converters for interfacing other industrial protocols with EtherCAT, can be developed.

2.5 FPGA

An FPGA is a digital IC that is fitted somewhere between PLDs and ASICs in terms of complexity and configurability. An FPGA contains configurable programmable logic blocks which are freely customizable by the design engineer, just as with PLDs but can contain a lot more logical gates. The FPGAs are still not as optimized in size and performance as an ASIC performing the same function. However, one of the drawbacks using an ASIC compared to an FPGA is the loss of flexibility and upgradability because the design is frozen in silicon once it is done. Some other drawbacks are the high price and the time consuming process of developing ASICs.

The flexibility of FPGAs is further enhanced by some vendors by introducing other systems or functions inside the same IC, such as processors, microcontrollers, RAM, DSP-units and multipliers etc. This gives the FPGA IC additional functionality in the same manner as a SoC IC. Some of these functions can also be added as vendor supplied or third party software IP-cores, such as the EtherCAT core used in this project. The FPGAs arrived in the market in the mid-1980 and were at the time a new and more complex product originating from the widely used CPLDs and PAL ICs. [p. 1-4, 8]

2.5.1 Overview of architecture and applications

FPGAs from different vendors usually have different underlying architecture but can all be programmed in at least the two most common used hardware programming languages, VHDL and Verilog. The architecture of FPGAs is not described in detail in this work but a short idea of how FPGAs work will be given next. All vendors use slightly different naming of their FPGA building blocks and internal functions, but the underlying idea is the same for them all.

In short, the buildings blocks of an FPGA consist of some sort(s) of LUTs, MUXs, SRAM and registers together with signal and clock connectors. These blocks are programmable to perform different functions and can be stacked or chained together to form bigger systems performing more complex functions. One FPGA can contain millions of these building blocks together with other logic, special functions or hardware described before. For more detailed information the reader is referred to the book in reference [p. 1-4, 8].

FPGAs are very widely used in different fields of application due to their wide suitability and broad complexity and cost range. The FPGAs are well suited in application fields requiring advanced parallel computations, a large number of I/Os and re-programmability etc. A few examples of application fields are DSP, ASIC prototyping, computer vision, medical imaging, cryptography, telephony centrals, industrial fieldbuses etc.

2.5.2 FPGA programming

The internal functions of the FPGAs are mainly written using programming languages such as VHDL, Verilog or System C. In this project however, VHDL was chosen as the programming language. As goes for the code itself, it can of course be produced with any text editor program available but each FPGA vendor offer their own design environment that is preferred to be used together with their FPGA. The design environment is needed at least when synthetizing the code and creating a configuration file for the target FPGA. Additional to this, each vendor usually offers different kinds of chargeable or free-of-charge software for design-aid such as IO-planning-, power optimization-, simulator- and IP-core generator software etc. to ease the design process. The vendors' own tools generate the configuration file from the hardware description written in VHDL, Verilog or System C and usually there is little or no need for the designer to interfere with this process. After the configuration file is written the user can download it to the physical device using a JTAG, a microcontroller or some memory device containing the file, such as an SPI FLASH.

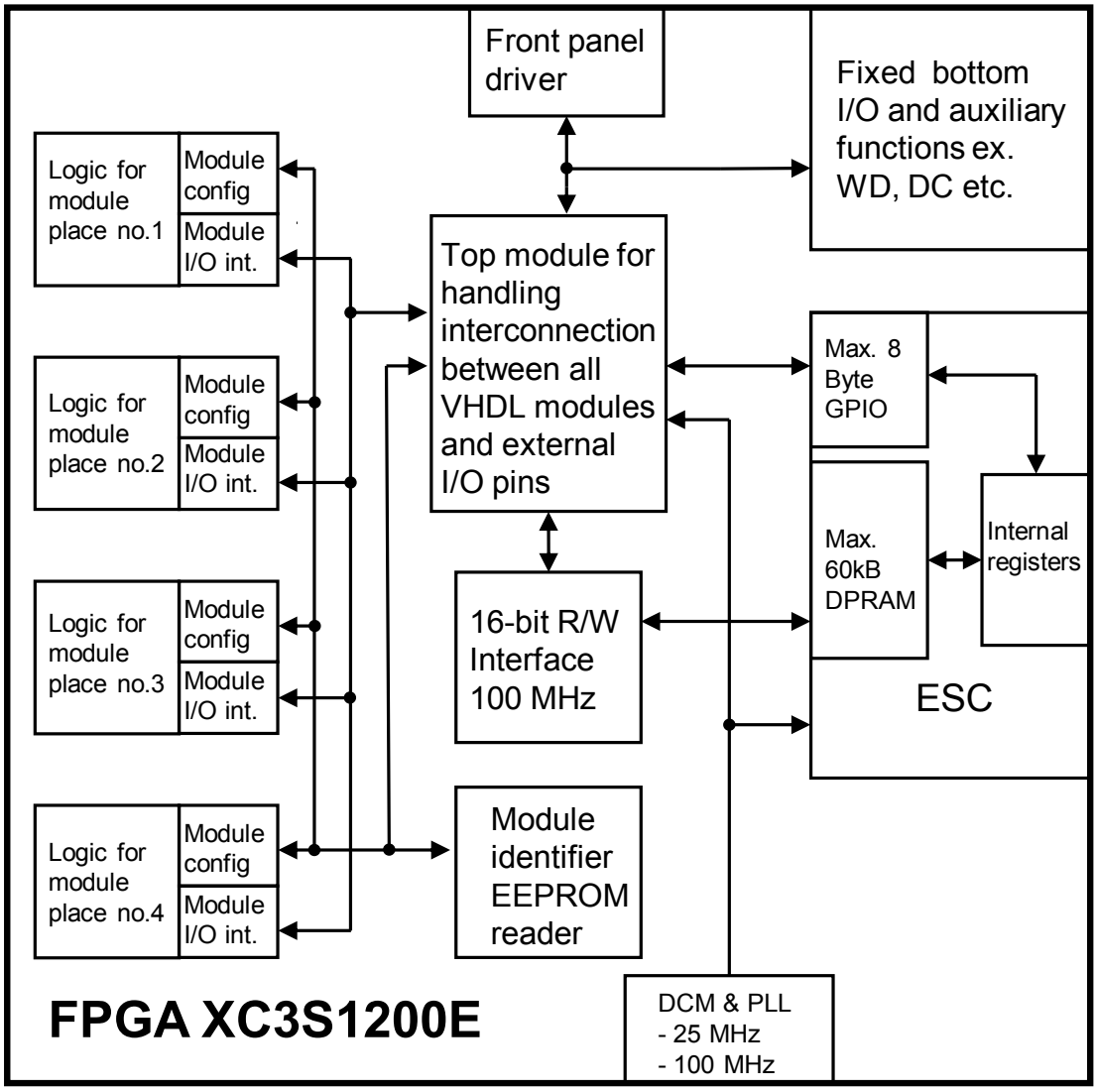
2.5.3 FPGA usage in the project fieldbus module

An FPGA suites this project very well because the module has a lot of complex functions that cannot be implemented with a PLD or microcontroller. A microcontroller cannot be used because the project involves a lot of fast switching I/O together with several simultaneous internal processes. Thus, the fastest microcontrollers would not be able to execute the instructions fast enough to satisfy the timing constraints in the processes. Due to the loss of upgradability and production volume size, an ASIC is not suitable for this project either.

The EtherCAT functionality is bought from a third party vendor (Beckhoff) as an encrypted IP-core that has to be inserted in the project inside the FPGA development tools. The IP-core is seen by the developer as a configurable black-box with visible I/O pins for interfacing. The design engineer can then add additional logic or other functions needed in the system design around this black-box.

Picture 4 on the next side shows the internal module blocks of the FPGA. In the center of the picture is the so called top module which is sort of a linking module, linking together the physical FPGA pins with the internal function blocks. It also links the internal modules together via signals and buses inside the FPGA. On the left side are four boxes labeled as logic for the optional modules. These functional blocks contain all the logic needed for all possible optional modules that can be connected. The system first reads the EEPROM of the option module connected using the module identifier block and then associates the correct logic with the option module and chooses the correct internal I/O going to the top module. In the top of the picture is the front panel driver which contains the logic needed for the front panel LCD, buttons and LEDs. The data is sent out to the front panel in serial mode to a daughter FPGA situated on the front panel PCB. In the right upper corner of the picture is the module that handles the 32 digital I/Os connected to the base PCB and handles some possible auxiliary functions such as the distributed clocks and watchdog, if used. Beneath this block is the Beckhoff EtherCAT IP-core with its internal registers and DPRAM. The DPRAM is connected to the other parts of the system via the 16-bit read/write µC interface, which controls the data flow to and from the EtherCAT DPRAM. The GPIOs of the EtherCAT module are as the name says general purpose not usually suitable as application I/O but can be used

with indicator LEDs and general buttons and switches. The last module of the picture is the digital clock manager module which together with a PLL generates the 100MHz internal clock from the 25MHz physical oscillator. The practical implementation inside the VHDL blocks in the picture will be dealt with in chapter 4.1.



Picture 4: Functional blocks inside the FPGA

3 Electronic and mechanical hardware design

This chapter starts with an overview of the mechanical design of the product, discussing both the design limitations and possibilities. After this, the electronic design is discussed and towards the end of the chapter there are two subsections discussing component selection and prototype testing and evaluation of the product. The component selection section discusses why the particular components were chosen and the prototype testing section discusses the different test phases and the observations made throughout the testing process. However, this chapter focuses mainly on the electronic design of the product, including both functional theory and PCB design, along with some discussion regarding the design related decisions that have been made in order to reduce costs, increase reliability and decrease noise etc. This section of the thesis focuses on the first near production-ready product as most of the electronic functions have been tested on breadboards and design-kits before this motherboard PCB was designed. The product is said to be near production-ready because naturally always some design errors and other surprises occur when designing a new complex PCB and testing it for the first time.

3.1 Mechanical design

The overall mechanical layout of the product including the PCBs, connectors and the plastic enclosure etc. is strictly limited to the same format as the other products in the product portfolio. Some small adjustments and modifications can be done, although the overall physical appearance has to be kept the same in order to fit all the parts together in the same manner inside the same enclosure as with the other products in the portfolio. As a result, the uniformity, both mechanical and visual, between the products is kept and the same plastic mold used for the enclosures can therefore be used for all the products in the family.

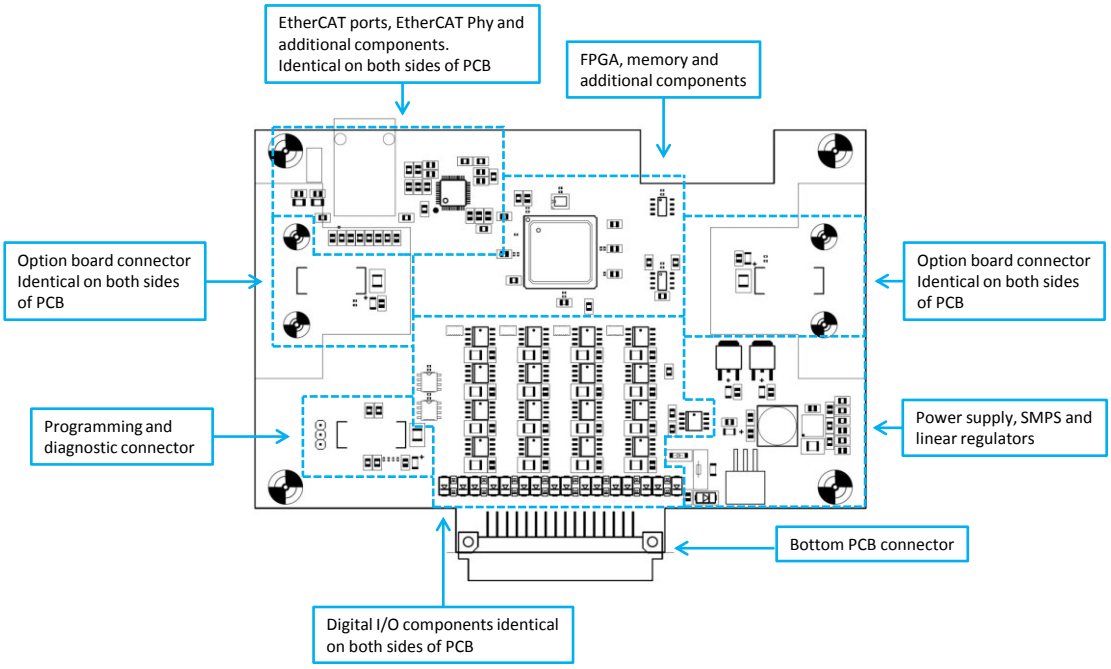
All the products in the portfolio also share the same type of bottom connector which is attached to the motherboard PCB and connects it with the bottom PCB. The bottom PCB is further on attached by screws to a metallic base plate, which for example can be fastened inside an electric cabinet. The bottom PCB of this product provides the motherboard with power and the physical connectors for the main part of the digital I/O. In Appendix A, the construction discussed is shown for illustrative purposes. Shown in the picture is the complete product, but not necessary exactly the final production version, inside the enclosure together with the bottom PCB and a preliminary front panel. The metallic base plate is not shown in the picture.

3.1.1 Module and motherboard PCB mechanical layout

As discussed in the previous section and shown in Appendix A, the motherboard PCB is fitted inside a plastic enclosure and therefore has to be of a specific size and shape. As a result, the enclosure sets the limits for all type of physical connectors to and from the motherboard and naturally also has an influence on the component placement to some extent.

In accordance to the enclosure layout, the four option module connectors have to be placed on both edges and on both sides of the PCB in order to practically get the physical connectors of the option modules outside the enclosure on both sides. The

picture in Appendix A. shows the option module placement on the enclosure and picture 5 below shows the connector placement on the motherboard PCB. Regarding the front panel, a small cut-out has to be done on the motherboard in order for an LCD display to fit. A connector with a flexible ribbon cable is used for the power and communication interface to the front panel. The EtherCAT RJ45-connectors are placed on the upper part of the PCB, accessible through holes in the front panel, to ease the EtherCAT wiring and to improve the visibility of the connectors' indicator lights. The PCB itself is held in place inside the enclosure by sockets, which go through the PCB's four mounting holes in the corners. The bottom connector is placed in the middle, on the lower edge of the PCB. The complete layout with some explanations can be seen in picture 5.



Picture 5. The complete layout of the motherboard PCB with some functional areas shown.

This picture will be further on referred to and explained in the section discussing the PCB layout and component placement.

3.2 Electronic design

The electronic design is next up now that the mechanical properties of the product have been discussed and examined. The electronic realization of the product is not nearly as limited with predefined restrictions as the mechanical. However, the mechanical limits also mirror themselves to the electronic design, at least to some extent. Some of these electrical predefined things are worth mentioning. First off, the overall physical measurements of the product naturally set limits to the component placement, number of components and to the overall size of the PCB. However, these limits are not difficult to overcome as the PCB is relatively large and almost all the logical functions are fitted inside one chip, which is the FPGA. Second, the input voltage to the product, which at the same time is the digital I/O voltage, is 24V with some tolerance. All the other needed voltages have to be derived from this voltage. Last, regarding the design in general, the product also has to comply with the international EMC and other standards concerning electronic products and therefore it has to be designed and tested accordingly.

In the next subsections the electronic design and component placement of the motherboard is divided into smaller functional blocks, which are discussed separately more in-depth.

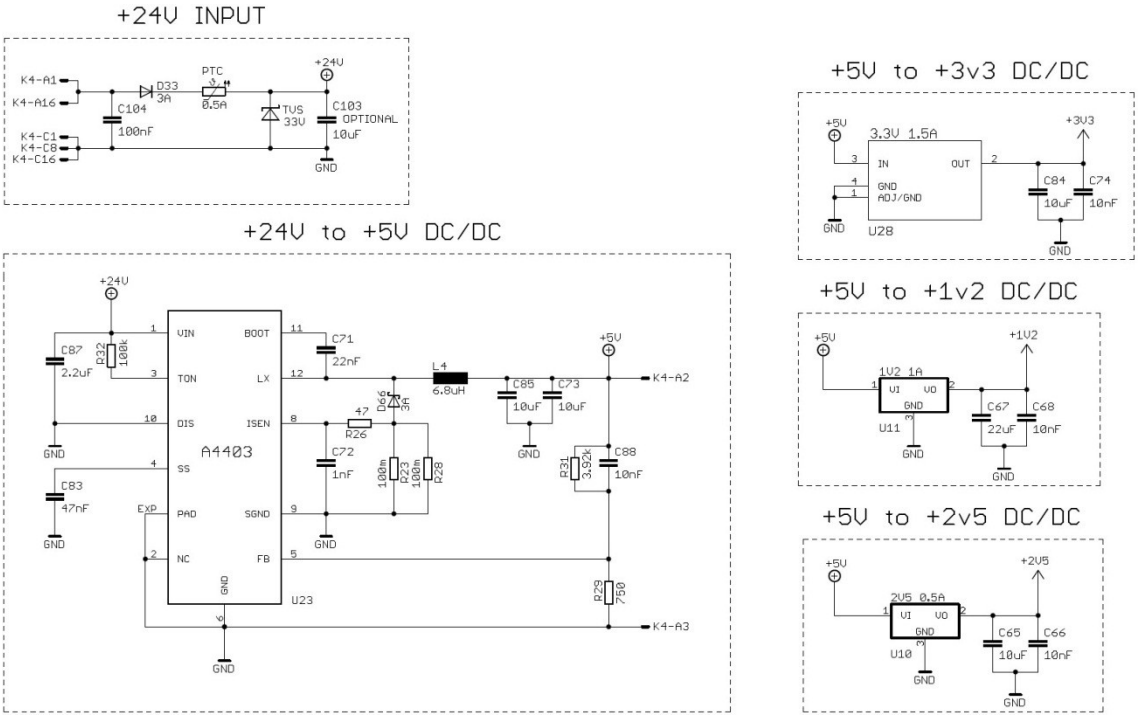
3.2.1 Power supply

As mentioned before, the motherboard itself is fed via the bottom PCB by a 24V power supply and from this voltage all the other necessary voltages have to be created. The motherboard needs a total of five different voltages in order to work. First is the 24V which is used by the bottom digital I/O and is also routed to the option board connectors. The second voltage is 5V, which is used by a small part of the ICs and all the linear voltage regulators in the design and further on fed to the option module connectors and front panel connector. The third one is 3.3V, which is used for the FPGA I/O banks, the EEPROM, the flash memory, the EtherCAT macs and also fed to the same connectors as the 5V. The fourth and fifth ones are 2.5V and 1.2V and are only used as the internal core and auxiliary voltages for the FPGA.

The power supply of the motherboard providing these voltages was designed to consist of one SMPS, which converts 24V to 5V, together with three linear regulators for each remaining voltage. This solution gives a very high efficiency ratio on the largest voltage drop together with a low-cost and low-noise solution on the lower voltages (3.3V, 2.5V and 1.2V). The two lowest can be made by two very small linear regulators because of the low current consumption. However, the 3.3V regulator has to be of a type capable of currents over one ampere because this voltage is used by most of the internal FPGA I/O, option board ICs, front panel, LEDs etc. This naturally results in more effort regarding the thermal design for the 3.3V than for the 2.5V and 1.2V. The thermal design considerations for these linear regulators as well as for the SMPS will be dealt with in the PCB section later on.

Next up will be a more in-depth discussion about the SMPS as it is the most complex part of the power supply. Beginning with the controller of the SMPS, an Allegro A4403 valley current mode control buck converter was chosen [9]. This part was chosen mainly because of the high switching frequency (smaller and cheaper passive components), high current output (up to 3A), wide input voltage range (9V to 46V) and minimum number of external components required. The controller schematic with

component selections was directly taken from one of the example schematics in the datasheet [p. 14, 9] for a 5V supply running at 1 MHz. The controller itself contains the switching FET, leaving the auxiliary components needed down to a schottky diode, an inductor and some filter and control capacitors and resistors. Faulty conditions such as overheating, over current, over voltage and under voltage etc. are directly detected by internal logic inside the controller and don't need any external arrangements. The complete schematic for the SMPS and linear regulators are shown in picture 6 below.



Picture 6. Schematics of the power supply

As seen in the picture, all the linear regulators are connected to the 5V supply and only accompanied by a few input and output capacitors working as filters, which is enough for proper operation. The upper left part of the picture shows the protection components of the 24V input to the module, which consists of a diode, a PTC, a Zener, a TVS and two filter capacitors.

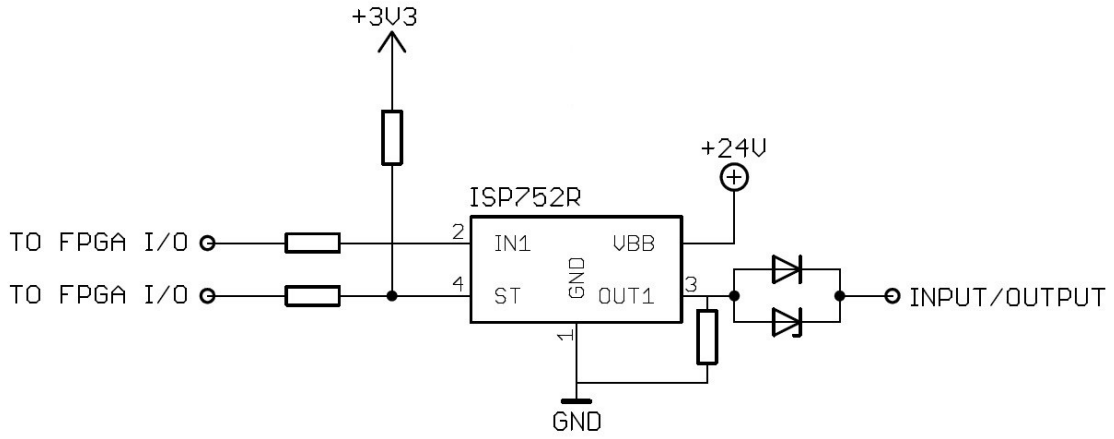
3.2.2 Bottom I/O and serial ports

The digital bottom I/O are the module's 32 24V I/Os with their physical screw terminals located on the bottom PCB and their respective electronic implementations on the motherboard, electrically connected via the bottom PCB connector. These I/Os are bi-directional, as will be explained later and consume the major part of the 48-pin bottom PCB connector. The I/Os are implemented using an ISP752R IC [10], which is a smart power high-side switch. This is basically an N-channel power FET with a CMOS compatible input and a diagnostic feedback, that are both connected to the FPGA via series resistors, for protection in case of failure. Further on, the ISP752R-switch also includes a lot of protective functions inside the chip, such as ESD protection, overvoltage detection, current limiting and thermal shutdown etc. The reader is encouraged to consult the datasheet for more information on these.

In order to get the same physical I/O pin to work in a bi-directional manner, as an input and an output, the diagnostic feedback feature of the ISP752R is used. The diagnostic

pin is an open-collector type and can therefore be read by an FPGA I/O pin using a suitable pull-up resistor. By using this configuration, the diagnostic pin will go low when a voltage is applied to the buffer output but no input is detected on the buffer input pin. This feature normally resembles a fault condition, indicating a short circuit to the supply voltage (24V), but is now used as the digital inputs in the module. According to the datasheet, the normal short-circuit-to-supply detection voltage is around 2.8V, which is a little low for an input-high signal when using 24V logic. This detection voltage is the voltage for which the diagnostic pin goes low, when a voltage is applied to the output at the same time as the input pin is low. To overcome this problem and raise the detection voltage a schottky diode parallel with a suitable zener was put in series with the output pin. By using this configuration the zener is raising the input detection voltage by the zener-voltage and the diode is providing a low voltage-drop path when the buffer is used as an output. A zener with a breakdown voltage of 9.1V was chosen in order to get the input threshold to be about 12V which is half of the 24V supply voltage. However, one important thing to keep in mind is the sensitivity of the input. As can be seen in the V_{bb} vs. R_O graph in the datasheet [p. 14, 10], the maximum thinkable output pull-down resistance would be about 300KOhm under extreme conditions. This value only needs about 10µA of reverse current to raise the output over the 2.8V threshold mentioned above. This current value is far too low to be used in practice because already a very small inductive or capacitive coupled noise current or even the diode reverse current alone would trigger the input. To overcome this problem, a load resistor with a suitable value is put parallel with the output pin to raise the input current to about 1mA @ 12V before the triggering happens.

When the digital inputs are implemented in this way, the normal diagnostic features when using the buffer as an output are not disrupted i.e. the diagnostic pin can be normally used to detect different fault conditions as described in the datasheet. The schematics of a single type of this digital I/O implementation are shown below in picture 7.



Picture 7. Implementation of one of the 32 digital I/O:s. The ST-pin (Status) is the diagnostic open collector output of the driver.

The motherboard is further on equipped with two RS-422 serial ports implemented by two high speed serial buffers connected to the FPGA. Both serial ports are fed through the bottom PCB connector to the base PCB. One of the ports is meant to be used for compatibility, namely to implement the old serial data protocol which was used with the older products in the ACN product portfolio. This feature is still optional and will only

be implemented if necessary. The other serial port is routed to a pinstripe and is primary left for future use.

3.2.3 Diagnostic, test and programming

Programming, reading diagnostics and testing the module in the production phase is done at some extent via a connector of the same type as the option module connectors. This connector is equipped with all the necessary signals for this purpose. The placement of the connector on the PCB can be seen in the lower left corner of picture 5.

For programming purposes the connector is equipped with the JTAG pins from the FPGA and two reset button signals. One button is used for resetting the FPGA and the other one used to reset only the EtherCAT functionality. The FPGA configuration memory is programmed via the JTAG bus indirectly through the FPGA and therefore no connection to the memory is needed on the connector. The FPGA will be factory programmed via this connector before shipping, while future firmware updates could possibly be loaded via EtherCAT.

For diagnostic and testing purposes the connector was further on populated with different signals. First, all the five different voltages used in the module are connected to their own respective pins for easier measuring while performing the initial tests of a new product. Second, two pins are populated with the SOF and EOF of the EtherCAT frame, used for diagnostic purposes. Last, some of the connector pins are connected directly to the FPGA I/O pins and are therefore firmware configurable. A small additional PCB, which is designed to be connected to the connector as a breakout card for the connector was manufactured, but it will not be discussed further in this thesis.

3.2.4 Option module connectors

As can be seen in picture 5, the four option module connectors are located at the exact same position on both sides of the PCB and on both edges. Therefore the connectors themselves have to be SMD components. The connectors are only accompanied with a few capacitors for each of the voltages connected to them. The connectors are fitted with 12 of the FPGA I/O pins each, used for data communication along with two FPGA pins each for the option board EEPROMs (more information about the EEPROMs in the software section) and a few pins each for the voltages 24V, 5V and 3.3V and ground. The maximum speed for the data transfer can be estimated to be about 2 MHz, which is low enough to avoid the need of any impedance matching resistors, as will be mathematically proven in the PCB design section. For the EEPROM signals, already a few kHz is enough.

3.2.5 Front panel connector

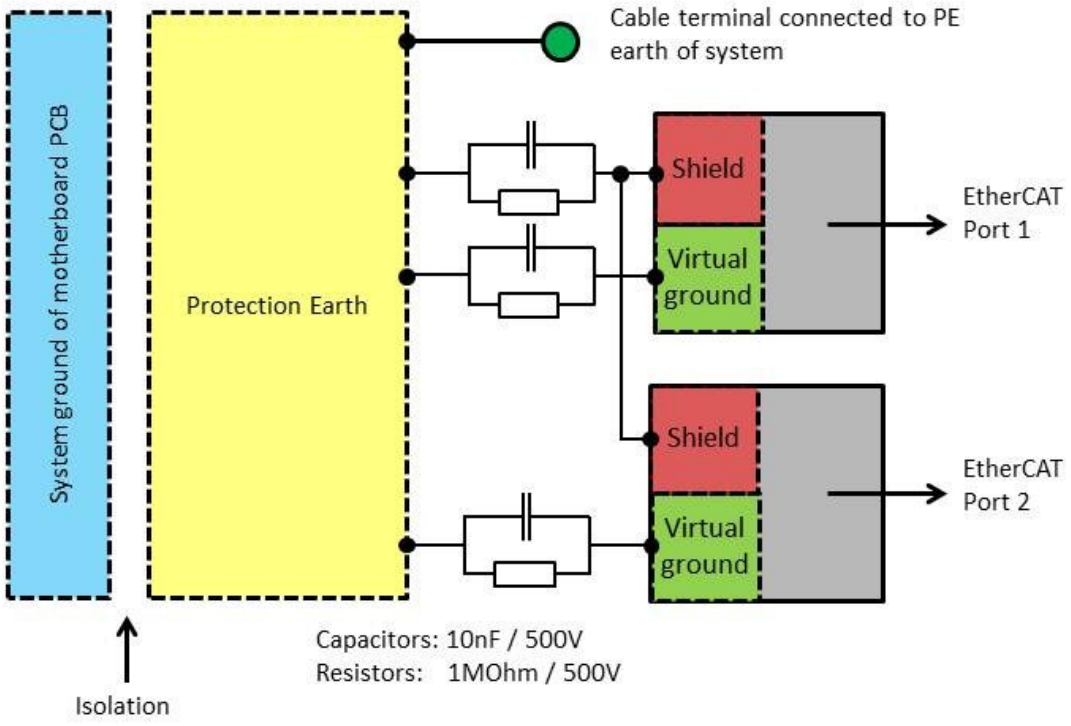
The place of the front panel connector is also shown in picture 5 and the connector itself is nothing more than an SMD-type ribbon cable connector, connecting the front panel with a flexible ribbon cable to the motherboard PCB. The connector is fitted with the 5V and 3.3V voltages and a small number of FPGA I/O pins for the data transfer. Because of the relatively low frequencies used (also estimated max. 2 MHz) by the data signals, no kind of signal termination are used in either end. If pull-ups or pull-downs are needed, these can be programmed inside the FPGA or put onto the front panel PCB. The data format is somewhat discussed in the software design section, but the front panel electrical and PCB implementation themselves are beyond the scope of this thesis.

3.2.6 EtherCAT ports

The RJ45 connectors of the two Ethernet ports along with the required components and Ethernet PHYs are situated on both sides, on the upper part of the PCB, as can be seen in picture 5. The two RJ45 connectors chosen for the motherboard are of SMD-type with integrated magnetics, which allows them to be placed on the same horizontal position on both sides of the PCB and reduces the overall component count. This connector type is capable of 10/100Mbit speeds (100Mbit is the requirement for the EtherCAT protocol) and only needs to be accompanied with a few pull-up resistors and a few filter capacitors. The connectors are also equipped with two LEDs, which in this product implementation are connected to the FPGA I/O and can therefore be freely programmed. [11]

Further on, the Ethernet MACs are connected to the RJ45 connectors directly using only the pull-up resistors and capacitors described above. The MACs themselves are connected directly to the FPGA I/O pins and most of the MACs configuration at startup (some of the I/O pins either at zero or supply voltage at startup) is mainly done by the Beckhoff EtherCAT IP-core and does not need any external components. The only things that need to be set by external resistors are the addresses of the respective MACs. As for the power supply, the MACs themselves usually need good filtering of the supply voltages and this was satisfied using a number of capacitors and serial inductors near the power supplies' pins. A reference connection schematic for the connections between the MACs and FPGA are shown in reference [III-79, 5].

To ensure proper grounding of the RJ45 connectors and the Ethernet connections overall, the design recommendation in reference [I-30, 5], was followed. In addition to this, a cable terminal connector was placed on the top side of the motherboard, meant to be used in cases when an Ethernet cable with a shield is used. The grounding arrangements are shown below in picture 8 and discussed on the next page.



Picture 8. Ethernet grounding

In the picture, it is seen that both shields and both virtual grounds are connected to the protective earth of the system via a parallel connected resistor and capacitor. This is done in order to ground high-frequency noise coming through the Ethernet cable shield and at the same time discharge static voltage buildup through the resistor. The reason why the shield is not directly connected to the PE-ground is to avoid ground loops in larger installations where different PE-grounds might be at different potentials. The protective earth on the motherboard PCB is isolated from the system ground in order to avoid inducing noise into the system. The high-frequency ground path is therefore made via the cable terminal to the PE ground of the installation, which usually is the backplane of an electric cabinet or a similar low impedance ground plane. It is important to keep this ground path as short as possible to avoid big ground loops of the shield, which are prone to inductive coupling. By using this grounding procedure in larger systems, the shields of the Ethernet cabling will be grounded for noise at each module in the system. The virtual grounds in the picture are the ground points of the transformer connections of the Ethernet connectors. These transformers and the virtual ground connections are physically completely situated inside the RJ45 connectors with only one pin leading to the outside world for grounding.

3.2.7 FPGA and memory

As could be seen in picture 5 earlier, the FPGA is placed centrally on the motherboard. This is not only done in order to get a short distance and easy routing to any surrounding ICs and connectors but also done in order to ease the surrounding capacitors' and resistors' placement around the FPGA, because especially the capacitors are critical to get closer to the FPGA power pins in order for them to work properly. The capacitors' physical sizes are decreased at the same time as the capacitors values are decreased. This is done in order to get the smallest value capacitors very near their respective power supply pins they are connected to. The placement for the smallest ones is usually on the flip side of the motherboard PCB, directly under the power supply pins which are connected via vias. The 25MHz oscillator is also placed very near the FPGA clock input pins to ensure proper signal integrity. The same oscillator output clock signal is also routed to the two Ethernet MACs, in order for the whole system to share the same clock. The FPGA configuration flash-memory uses the highest frequency of any signal going outside the FPGA (75 MHz) and so is impedance matched for signal integrity using a series resistor. In contrast, the EtherCAT EEPROM is directly connected because of the much lower speed (Max. 400 kHz).

The FPGA itself can be programmed in a number of ways depending on factors such as the FPGA configuration memory type used, JTAG connection, startup configuration device (memory or JTAG) and possible FPGA chaining etc. The configuration mode in this module is set in hardware on PCB level by fixing each configuration pins to its required potential, which is either supply voltage or ground. The configuration used in this design allows for a high speed (75MHz) cheap serial flash memory to be used as the FPGA configuration memory and allows it to be programmed indirectly by the FPGA via the JTAG port. The FPGA uses three different supply voltages, for which the 3.3V is used as the I/O supply voltage, the 2.5V and 1.2V as the internal core and auxiliary voltages. All the power pins of each voltage are spread throughout the pin grid of the FPGA BGA package, which means that the capacitors for each voltage can be easily placed evenly around the FPGA perimeter (the bigger capacitors) and on the flip side of the PCB behind the FPGA (the smaller capacitors). For more information regarding the FPGA, consult the datasheet found in [12].

The two memories used in the module are both of serial type (SPI for the FPGA configuration memory and I^2C for the EtherCAT EEPROM memory) and are both compatible directly with the FPGA 3.3V logic. The minimum sizes of the memories are dependent on the FPGA logic size and the EtherCAT minimum memory requirements (the ESI is stored in the EtherCAT EEPROM). Neither one of the memories is locked by the FPGA, which means that a larger memory than needed can be used if for some reason a non-volatile storage device using the free memory sectors, is needed.

3.3 PCB design

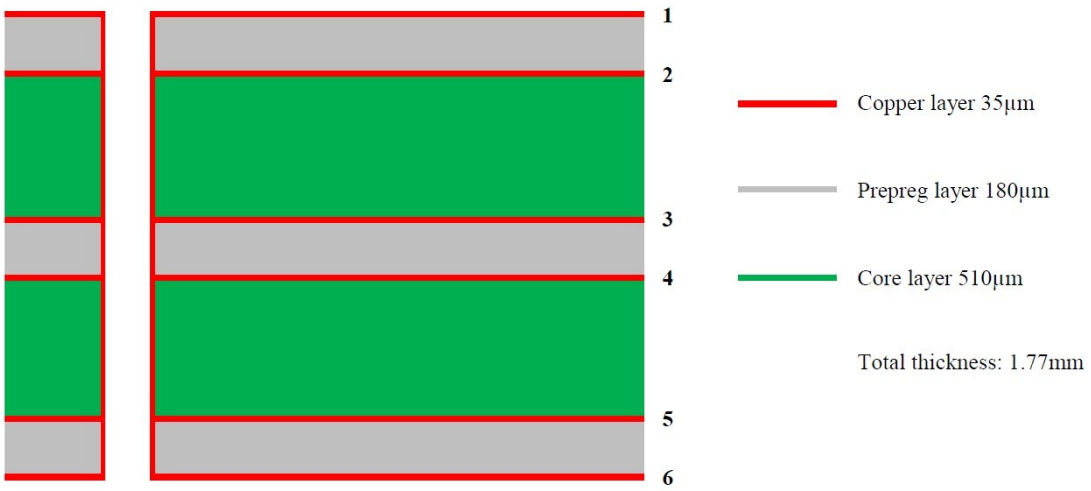
The design process of the motherboard PCB was started after the complete electrical schematic, discussed in the previous section, was completed and the physical measurements of the PCB were defined as discussed in section 3.1. The PCB design process in this project can roughly be divided into five different stages, which are numbered below for a clearer overview.

1. Define the physical measurements of the PCB board, the stack-up and set the placement of all screw holes.
2. Place, fit and group all the components on the board.
3. Route all the signals and define the ground and supply layers.
4. Design and draw the top and bottom silk-screen layers of the PCB.
5. Go through steps 1-4 again and look for errors before manufacturing the first PCB prototype.

In the following subsections, all of these steps except step 2, which was discussed in the previous section, will be examined further and explanations will be given to the solutions that were made. In the last subsection a few words about manufacturing the PCB will be given along with some discussion about the factors that influence the PCB price. The motherboard was designed using the Cadsoft Eagle PCB design software [13] and the first version was manufactured by the Finnish PCB manufacturer Elprintta Oy [14].

3.3.1 PCB board stack-up and properties

The outline i.e. the physical measurements of the PCB were already shown in the previous section in picture 5 and therefore this section will deal with the properties of the board. The PCB chosen for this project is a board with six layers with vias going from the top layer to the bottom, as can be seen below in picture 9. The fact that the vias are always going through the whole board, when used, is a PCB manufacture process limitation. The copper thickness of all layers are the standard 35µm and the prepreg and core layer thicknesses were chosen from a list given by the manufacturer and they were designed to add up to a total thickness of about 1.8mm. The PCB board maximum thickness is set by the plastic enclosure's measurements. The board had to be made as thick as possible in order for the Ethernet RJ45 connectors' plastic feet to fit inside the same hole from both sides of the PCB.



Layer	Function
1	Signal
2	GND
3	Signal & GND fill
4	Signal & GND fill
5	3.3V
6	Signal

Picture 9. The motherboard PCB stack-up

In order to get most of the pins from the FPGA 320-pin BGA package routed easily throughout the board, the PCB layer count is set to six. The routing is also affected by the manufacturer's recommendation of a minimum drill size of 0.2mm, which was the smallest possible for this kind of design without having to use micro vias, which would have made the board much more expensive. Another recommendation is that the minimum trace width and isolation clearance is equal or larger than 0.15mm.

To add some stability when pressing the multilayer PCB together, the inner signal layers were ground-filled in order to get their thickness to be uniform. A further advantage of the ground filling is that it adds more ground layers to the PCB, which are quite uniform because the inner signal layers were only routed with relatively few signals. All the ground layers are further on tightly connected together with vias which are distributed evenly throughout the board. This strengthens the thermal coupling and equalizes possible potential differences. As can be seen in picture 9, two layers were completely reserved for the supply voltage 3.3V and ground. The voltage 3.3V was chosen as the supply layer voltage because most of the ICs used are either 3.3V devices or have 3.3V tolerant I/Os. The 2.5V and 1.2V supply voltages were also routed in the same 3.3V layer. The layers, especially the ground layer, significantly increase the heat transfer from the linear regulators and the SMPS. As a result, little design effort is needed in order to ensure proper cooling of the devices and no add-on heat sinks will be necessary.

As could already be seen in picture 5; the SMPS together with the linear regulators are placed in the right lower corner of the PCB. This is done because it is the shortest distance to the PCB bottom connector's supply voltage and ground pins. The ground layers under the power supply components are also isolated from the rest of the system's ground and only a small strip just above the bottom connector's ground pin connects the two ground areas together. This is done in order for the noise originating

from the switching to go straight to the module ground without interfering with other parts on the PCB.

Last in the design chain to be implemented was the silk screen of the PCB. All the component names and markings about orientation were added to this layer and were put out on the board in a manner for them to be visible also when the board is completely populated with components.

As a PCB design reference for multilayer boards, reference [p. 109-166, 15] was referred to frequently throughout the design process. As for the stack-up, the reference stack-up for a six-layer board, figure 6-34a in [p. 149, 15], with four signal layers was used. In the reference, it is stated that the signals of adjacent layers should be routed perpendicular to each other. Although, this was found to be very difficult to implement and therefore it was ignored and instead the ground fill of the inner signal layers was used to isolate possible capacitive coupling between the signals of adjacent layers.

3.3.2 Routing

The process of routing the PCB was started out by first routing the most critical signals, in terms of length or width. The most critical signals with respect to length would be all the clock signals, high-speed data transfer to the PHYs and the FPGA memory data signal. These were routed in the straightest possible way in the top layer, because of the thin isolation (0,18mm prepreg) to the ground layer beneath; while attention was paid that the ground plane was continuous beneath the signals. In the same manner, considering the widths of the traces, the most critical signals are not only the power supply ones, which carry the largest currents, but also all the ground signals which are not directly connected to the ground plane via vias.

After this, the I/O signals originating from the FPGA BGA package pins were routed outside the package perimeter using all the four signal layers, as these are the most difficult signals to route. All of the non-special-function FPGA I/O pins can be routed arbitrarily to their respective connections because the FPGA configuration software can normally route any internal signal to any package pin. Therefore it is up to the designer to choose which pin is associated with which I/O function. The special-function signals in this case mean I/Os with any special functions, such as clock inputs or reference voltage inputs etc. If one uses these special functions, they naturally have to be routed to the respective pin with the feature in question. Since the FPGA was placed near the middle of the PCB, the signal routing is somewhat easier in all directions and the trace lengths can be kept moderate with all connections. The minimum trace width and isolation however is set by the manufacturer's recommendation (0.15mm), which is enough to get the FPGA properly routed. The maximum trace widths vs. current were checked with a built-in tool in the Cadsoft Eagle software. The maximum current for the I/O signals traces was reported to be 0.64A, which is more than enough because no high-current devices are driven directly by the I/O pins. For the frequency response vs. trace length the design rule mentioned in reference [p. 142-143, 15] was used. It states that the critical length for which the trace can be seen as electrically short is measured from the propagation delay and is such that the propagation time is less than one half of the rise-time. Calculations with values from the motherboard PCB are explained and shown below in formula (1).

$$t_{PD} = \frac{1}{2.54} * 84.75\sqrt{0.475 * 4.8 + 0.67}\,\frac{ps}{cm} * 15.23cm = 0.873ns \quad (1)$$

$$0.873\,ns < 0.5 * 4.06 = 2.03ns$$

The fastest rise-time on the I/O pins achievable for this FPGA was found in the datasheet to be 4.06ns while using maximum drive current together with 3.3V logic levels. To calculate the propagation time for the trace, the formula for t_{PD} in [p. 154, 15] was used, with an $\epsilon_R = 4.8$ for normal FR4 PCB material. The longest FPGA trace is 15.23cm, which is one of the JTAG traces. As can be seen from the result, 0.873ns is much smaller than half of the fastest rise-time possible with the FPGA. Therefore no termination resistors will be needed. Additionally, there is no need to even use the fastest rise-times for the I/O pins in the design, which gives even a bigger margin. Another important thing to keep in mind is to keep the routing of the signals without steep corners in the traces, because they cause sudden impedance changes, such as explained in [p. 150-152, 15]. Neither are too wide corners good because they make the PCB manufacturing process harder, as is explained on one of the PCB manufacturer's homepage [16].

As goes for the routing in general, not only proper measures for signal integrity have to be taken into account, but also the PCB manufacturers' design rules have to be obeyed in order for them to produce a reliable PCB board. Some of the manufacturers' general PCB design rules were taken from [16] and some asked directly from the manufacturer, such as the minimum trace width, the minimum isolation distances and drill sizes. These factors depend on the manufacturers' processes. However, the PCB of the motherboard were designed in a manner that would make it possible for it to be manufactured by as many vendors as possible, while still providing a good PCB design result satisfying all the requirements.

3.3.3 Manufacturing the PCB

Before manufacturing the PCB a number of documents including plotter files and other data, have to be produced and sent to the manufacturer. The Eagle Cad software produces these files directly from the PCB board layout editor while the user only chooses which design layers to include. These files include all the data of the board such as, plotter data for the signal layers, silk-screen data, drill-data and stencil mask data etc. An interesting matter of manufacturing the PCB is to investigate which parameters affect the manufacturing costs the most. It can be found, by asking the manufacturers, that the cost is mostly defined by the number of layers the PCB consists of, if the PCB size is kept constant. The second most influential parameter is the number of PCBs that are ordered at the same time. Third is the delivery time, for which the costs can be reduced up to about -30% by adding a few weeks to the delivery. Other factors that play a moderate to small role are the minimum trace widths & isolations, the minimum drill size, the number of holes and pad coating material. As a conclusion, when keeping the board size constant as in this case, the only thing that plays a significant role at the design phase is to keep the layer count at the minimum required amount as the other factors such as the PCB delivery time and delivery amount are more of a concern of production management.

3.4 Component selection

This section will discuss the component selections in the design, concerning each functional block and all the different types of components used on the motherboard. Factors such as availability, irreplaceability, life-length, cost, values and size etc. will be covered. Although, these factors will not be discussed in-depth for every part type, but rather only some relevant information will be given. The following subsections are

divided into active components (ICs), passive (resistors, capacitors, inductors) and connectors.

3.4.1 Active components - ICs

First up of the ICs is the FPGA, for which Xilinx was chosen instead of Altera as the manufacturer because of Xilinx's big market share, which usually stands for better product support and more users encountering the same problems etc. Another essential reason why the FPGA has to be from either Xilinx or Altera is that the encrypted Beckhoff EtherCAT IP-core is compatible with only these two manufacturers' FPGAs. As for the specific FPGA that was chosen from a broad product family, some critical parameters had to be met, such as the maximum speed (at least 100MHz for the EtherCAT core), I/O pins (a minimum of 200), logic enough for EtherCAT core + microcontroller and other logic. The parts that meet these specifications are either the 1200E or the 1600E from the FPGA family given in [p. 2, 12]. The software for the FPGA was not ready when the part was chosen which means that the exact required amount of logic was not known and therefore the software development was started with the largest FPGA of the family (1600E, which surely has more logic than required) and then will be downgraded later on in the production versions. The configuration memory for the FPGA was chosen from the memory compatibility list in table 56 [p.79, 12] and the EtherCAT memory was chosen to be of the same type as used in the EtherCAT evaluation board, to ensure that no compatibility problems will occur.

The Ethernet controllers have to be chosen from a list, managed by Beckhoff [17], of EtherCAT compatible MACs in order to ensure that they work together with the Beckhoff IP-core and the EtherCAT protocol. The type chosen for the motherboard is given in reference [18] and is chosen to be of the same type as used in the CPU module of the same product family.

For the bottom I/O, the intelligent proFETs used as buffers were chosen because of their large current driving capability for all kinds of loads (inductive, passive etc.), without the need for many external components, such as protection diodes or equivalent. Some other advantages already partly discussed before are that they can be used as inputs and have a comprehensive fault diagnostic system built-in inside the chip. After some investigation it was also found that larger manufacturers of digital I/Os use the same components, which usually means that they are well tested and there is a lower risk, due to a larger demand that the production will end in the near future.

As goes for the two RS-422 serial ports, dual line drivers of type UA9638 and dual line receivers of type UA9637 were chosen. The choice was made due to that they both can exceed 1,25Mbit/s transfer rate (EIA-422-B standard), which is the maximum speed that will be needed in the product. The 1,25Mbit/s value comes from the fact that older CPU units in the ACN product portfolio used this speed for communication and therefore it will be possible to use them together with this new ACN I/O-module. All sorts of serial communication with a PC or other conceivable devices to connect to the ACN I/O are often possible and practical already at much lower speeds.

Moving to the power supply part, the SMPS controller found in reference [9] was chosen because of the high frequency (smaller components), large current capability and minimal need of external components. The current feed capability is much higher than required by the motherboard alone. This current reservation was made in case for possible future use, for example if some upcoming option boards require larger current feed capabilities. As goes for the linear regulators, one criterion was that their ground

terminal should be connected to the component base in order for them to share the same copper-plane heat sink on the PCB. Another criterion only for the 3.3V regulator is that it has to have larger current capabilities than the smaller ones partly because of the same reason as with the SMPS and partly because the FPGA I/O pins use the 3.3V supply voltage as their logic-high level.

3.4.2 Passive components - capacitors, resistors and inductors

Common for all of these three component types is that most of these components were chosen to be SMD components of size 0805. This choice was made due to that they are easily soldered by hand and very cost efficient independently of value and/or voltage tolerance. Some exceptions were made and they will be explained in the next paragraphs of this section. Another general design rule was using as much as possible same value resistors and capacitors throughout the design whenever possible, in order to decrease the number of different components.

Most of the resistors on the motherboard are used as pull-ups, pull-downs and protection or termination resistors. This fact eases up the resistors' resistance tolerance and maximum voltage tolerance and therefore cheaper types of resistors can be used almost throughout the whole design. Only a few exceptions exist and they are the pull-ups of the RJ45 connectors, the current sense resistors of the SMPS and the Ethernet cable grounding resistors. The first two has to have a more precise resistance tolerance and the last ones have to have a voltage tolerance up to 500V.

The capacitors used in the design were chosen to be either tantalum or multilayer ceramic capacitors. One reason for this choice is that these types of capacitors have the largest frequency span for which they work properly. The ceramic capacitors where further on chosen to be of the types X5R or X7R for the ICs which operate at higher speeds and of type Y5V for the slow bottom I/O drivers. The reason why not all the capacitors are of type X7R is that it would significantly increase the component costs, at least for the larger valued capacitors used with the slow I/Os. The capacitors size used were all of size 0805 except for the ones situated behind the FPGA and the ones used together with the Ethernet grounding (500V tolerant, similar to the resistors discussed earlier). Behind the FPGA, the SMD size was reduced to 0402 because no larger components would physically fit between all the vias coming from the BGA package. It is not reasonable to place them outside the BGA perimeter either, because then the trace lengths would become too long for the smallest 1nF capacitors to work properly.

There are only a few inductors used in the design and they are all, except one, coupled to the different power supply pins of the Ethernets' macs for filtering purposes. [p. 25, 18] shows the recommended reference design for the power supply and it was also directly implemented in the motherboard design. The single inductor left is the one used in the 5V output of the SMPS. The specifications for this inductor were that it should be suitable for the high switching frequency and that the saturation current would be high enough for it not to saturate in extreme load conditions. The inductor type specified for this design was an encapsulated toroid-inductor. This type is well shielded from the surroundings and has good current capabilities already at small physical sizes.

3.4.3 Connectors

The RJ45 connectors were chosen to be of SMD-type in order to fit the two connectors at the same position on opposite sides of the motherboard PCB. This positioning was done in order to save space on the front panel PCB. The connectors for the option boards were also chosen to be of SMD-type for the same reason to be able to place them on opposite sides of the motherboard on the same position. This placement strategy makes it possible to connect any option board on any of the four option board connectors. Because the option boards won't normally be taken away while they are once installed, the SMD-type of connector is good enough although it is much more fragile than a through-hole connector. Another SMD connector type used on the motherboard is a dense flat SMD ribbon-cable connector for the front panel. This connector was chosen just in order to save space. The only through-hole connector used in the design is the bottom PCB connector, which naturally has to be more robust.

3.5 Evaluation and testing

Testing and evaluating the functionality of the first prototype is also a many phased process, like the design process itself. In the following subsections the different phases will be gone through and the results and observations made along the process will be discussed. Lastly, some future testing procedures will be discussed.

3.5.1 Test phases

Testing the prototype was naturally started with a visual inspection of the PCB board. In a multilayer board it is impossible to see every signal layer and connection but an overall picture of the manufacturing success can be seen. No visible errors were found, therefore the PCB was sent to a component assembler in order to solder the components that could not be soldered easily by hand, such as the FPGA BGA package and the SMPS controller QFN package.

The second test-phase was soldering the motherboard power supply components on the PCB. This phase included checking all the voltage levels and polarities, the 24V supply current and component heating. Two important things that were left untested in this phase but will be tested later on were the power supply transient response and the voltage fluctuations on each voltage with varying loads. This is done better later on with all the components assembled and the FPGA programmed with the main part of the functionality. Some non-fatal design errors were observed in this phase. The least serious problem was that some of the component packages were of a little different physical size than the corresponding parts in the Eagle CAD software, although this didn't disturb the soldering process. Another more serious thing that was observed was that the 3.3V and 1.2V linear regulators did not have their ground terminal connected to the component body as was expected. This problem can be overcome by changing the regulator type to another.

The third test-phase included soldering all the FPGA related components, such as the capacitors, resistors, memories and programming connectors. The intention in this phase was to test whether a connection can be made to the FPGA via the programming JTAG-cable and whether the FPGA configuration memory can be written and read. The test results showed that all things worked as expected and no glitches or other abnormalities where observed after a few test cycles.

The intention of the fourth test-phase was to test whether the two Ethernet ports worked correctly when using the EtherCAT protocol. This phase included soldering the Ethernet PHY components as well as programming the EtherCAT IP-core inside the FPGA. The test results showed again that everything worked as expected except for one of the LEDs on the Ethernet ports which was connected in reverse. More information on programming and configuring the IP-core is given in the software design section.

In the fifth, which was also the last prototype test-phase, all the drivers for the 32 bottom I/Os were soldered along with the option board connectors, front panel connector and the RS422 transceivers. The option board connectors were tested with a simple expansion I/O option board (an I/O board with same type of I/Os as the motherboard bottom I/Os) and were found to be working. The data transfer through the front panel connector was tested with a front panel PCB and also this worked as expected. All the outputs of the bottom I/Os worked, but it was found that the inputs needed additional resistors to pull up the open-collector status pins, which are used as the input. Also the RS422 transceivers' communications were tested with a serial cable to a PC. No problems were found with the communication.

More testing will be conducted before the product is ready for production. The upcoming tests will be done by placing the product in real operating environments. This will put more types of stresses on the product, such as heat stress (product inside a closed cabinet), EMC and other disturbances stress (product close to a frequency inverter and conducted disturbances coming through data cables, such as encoder cables) and shaking stress (product placed near or on shaking machinery). At the time this is written, however, the product is not ready for these types of tests and therefore no results can be given.

4 Software design

This section of the thesis will deal with the software design of the module by mainly focusing on the VHDL-language hardware design and to a smaller extent discuss some of the additional languages' role in the design, including languages such as Assembly, microcontroller specific C and the XML markup language. Some of the software design was already initially begun in the early prototyping stage when all the functional parts were tested on breadboards and evaluation kits. Even though the target FPGA on the motherboard is different from the evaluation kit's FPGA, main parts of the code can be directly imported and implemented with the motherboard's chip. This will speed up the initial software development and testing. However, the remaining and largest part of the code was still unwritten when the first prototype PCB was ready because a large part of the features would have been hard to test properly without the motherboard prototype at hand.

The section will start off with the VHDL-design part and begin by discussing some basics of VHDL software design flow. After this, the FPGA software hierarchy and design used in the product will be discussed as a whole and then further on be divided into smaller functional blocks. In the end of the first part, some words will be said about the design tools used together with the FPGA. Also some future features not yet done at the time of this writing will be discussed. The second part of this section will begin by discussing the soft-core microcontroller and its own internal software and later on discuss the XML markup language used by the EtherCAT master. The last part of this section will discuss software related testing and evaluation.

4.1 VHDL software design

The VHDL language originates from a VHSIC program set up by the United States government in the early 1980's. During the program, a need for a common description language for ICs arose and this finally resulted in the first standardized version of the VHDL language, VHDL-87. VHDL borrows heavily from the early programming language ADA and since the beginning, many new revisions of the language have been standardized, with the newest one being VHDL-2008. The VHDL language does not only fill the need to describe the structure of a system and its connections to subsystems using familiar programming language forms, but also allows the designer to simulate the design, using the same VHDL language, before producing the final product. [p. xvii – xviii, 19].

Although conventional programming languages such as C, Basic, Pascal etc. are used for completely different purposes than hardware description languages some comparisons can be made. First off, there are some big differences between the conventional programming languages and VHDL, by the most prominent one being that VHDL is a parallel language and the others are sequential languages. This means that statements in VHDL are executed concurrently (in cases where the small internal hardware delays are ignored) versus statements in for example C, which are always executed in their own order sequentially.

Because VHDL is also a language capable to be used in simulations, not all of the VHDL language is synthesizable in hardware. This means that the coder has to pay attention to use only code that can be implemented by the specific design tools (Xilinx

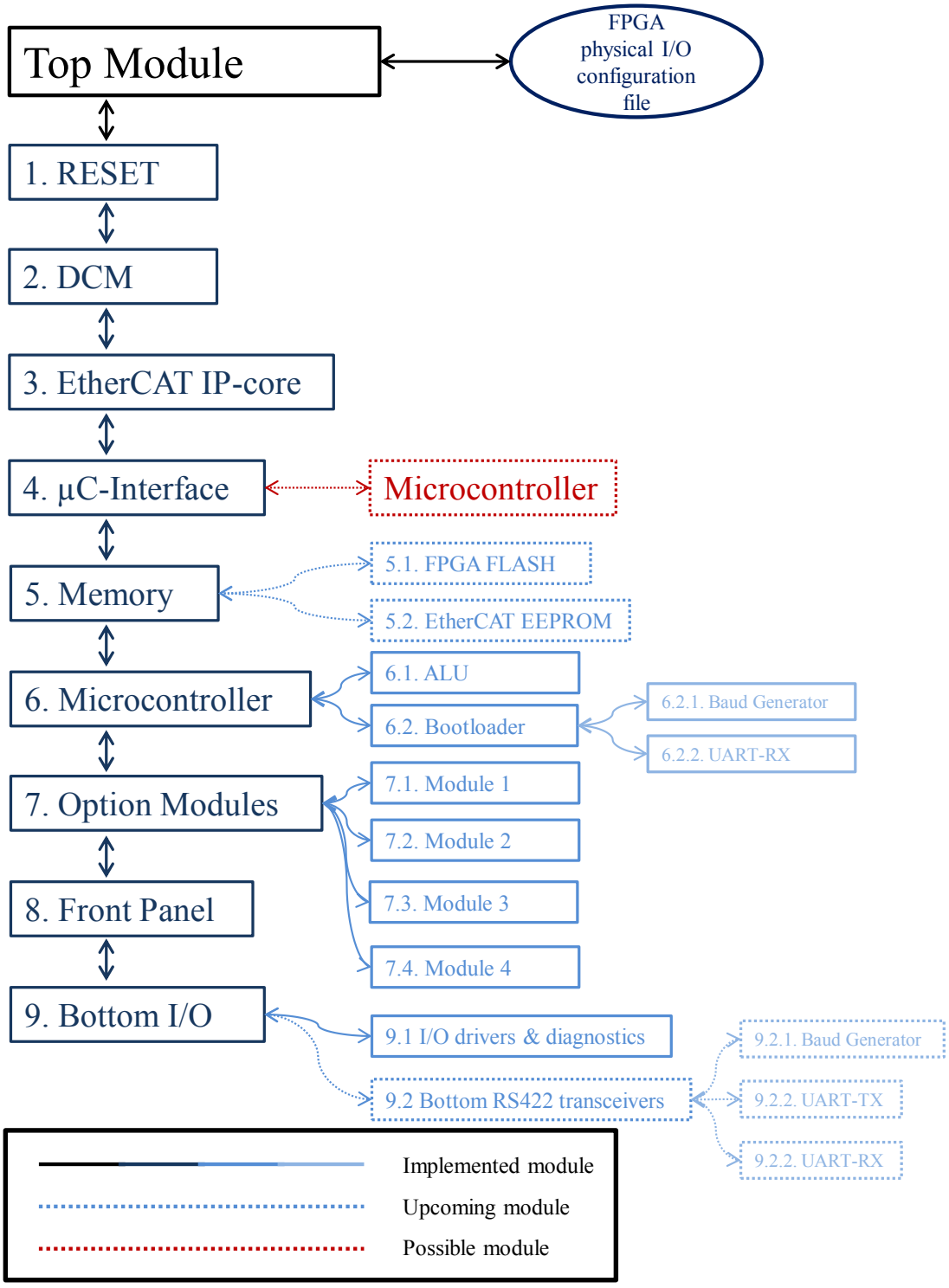
in this project) on the specific FPGA used. As a good guideline for this, reference [p. 633-668, 19] was referred to throughout the project.

4.1.1 The overall structure of the VHDL description

The complete VHDL description of the FPGA, including all its different internal functional blocks are divided into several VHDL description files and are all instantiated and linked together inside a so called top module. The top module is also a VHDL file itself, but it doesn't contain any higher level logical functionality or other signal manipulations. The module's purpose is only to link the several VHDL files of the project together in the right manner (signal connections) and provide a description for the compiler about the connections between the internal logic and the physical I/O pins of the FPGA. All of the additional VHDL files containing the different logical functionalities are instantiated in the top module and seen here as only black-boxes with port maps describing the signals coming in and going out of the modules. This design approach was chosen because it gives a good overview of the complex design with all the interconnections between the different functional blocks. It also allows for all the VHDL functional modules to be worked on individually and to be easily instantiated several times, if needed. Further on, it allows the design to easily adopt new VHDL modules by adding VHDL files to the project, instantiating them and connecting them together with the other modules inside the top module, while keeping the old module hierarchy intact if wanted. The encrypted Beckhoff EtherCAT IP-core module is instantiated inside the top module in the exact same way as with all the other functional blocks. The next two subsections will discuss the module hierarchy of the design and further on shortly describe each module's individual functionality, without going into specific details.

4.1.2 The complete VHDL module hierarchy

From top to bottom, the hierarchy of the VHDL configuration program starts out with the top module described in the previous section and ends in several branches containing VHDL description files which contain smaller blocks of logical functions. The hierarchy was attempted to be kept four levels deep at maximum, as a good compromise between a clean code structure and number of branches. Picture 10 on the next page shows the complete hierarchical code-structure, along with the names of the blocks. In the picture, the module connections are shown from top to bottom and from left to right. The FPGA physical I/O pin configuration is directly connected to the top module together with all the other modules branching downwards, as can be seen in the picture. The red-dotted area is a possible upcoming microcontroller module as a replacement for the VHDL-implemented µC-interface. This module will be discussed in the next section. The blue-dotted areas are future modules not yet implemented at the time of this writing. The first two are interfaces to the two memories on-board, which could be used to store non-volatile parameters and programs etc. The rest of the blue-dotted boxes are intended for the RS422 transceivers connected to the bottom PCB. All the non-dotted boxes are implemented in the first production version of the product and they will be explained more throughout in the next subsection.



Picture 10. Overall view of VHDL program hierarchy

4.1.3 Functional descriptions of the different VHDL modules

The next thing up is a description of the modules in picture 10. The complete VHDL implementation of the modules which are discussed here are not given due to company secretes but despite this fact some interesting and important features can be discussed. The most interesting module combination which can be discussed more in-depth is the microcontroller module together with its surrounding modules (Memory, Bootloader and ALU) which in turn corresponds to the modules 5.xx-6.xx in picture 10. The rest of the modules will be covered more briefly in the text.

Modules 1 to 4 and 9

The first module in picture 10 is the reset module of the whole system. This module is intended to handle all the reset signals and reset conditions of the system. This includes holding the module in reset before the digital clock managers have locked and resetting appropriate functions inside the FPGA accordingly to different reset conditions such as external reset triggering, internal reset etc.

The module number two is the DCM or digital clock manager module for the system. This module takes the external oscillator (25MHz) as an input and from this generates three different frequencies which are all in the same phase. The frequencies are 100 MHz, which is used by the EtherCAT functionality, 25MHz which is used by the microcontroller, memory and option modules and finally 5MHz which is used by all other low-frequency functionality such as the front panel unit and the bottom I/O.

The third module is the encrypted Beckhoff EtherCAT IP-core. Because of the encryption, this module only show the port map and some basic configurations to the user and therefore the only thing the designer can do is connect the appropriate signals to the ports and configure things such as vendor ID, product number etc. However, these configurations are primarily done automatically with software that Beckhoff included with the IP-core.

The fourth module is the µC-interface, which lies between the user logic and the EtherCAT IP-core. This interface exchanges data to and from the EtherCAT packages and so functions as the link between the local application on the ACN I/O module and the EtherCAT system. In upcoming versions this module, now purely implemented in VHDL, is probably replaced by a VHDL microcontroller in order to add some functionality that is now left out. This will not be discussed further in this thesis.

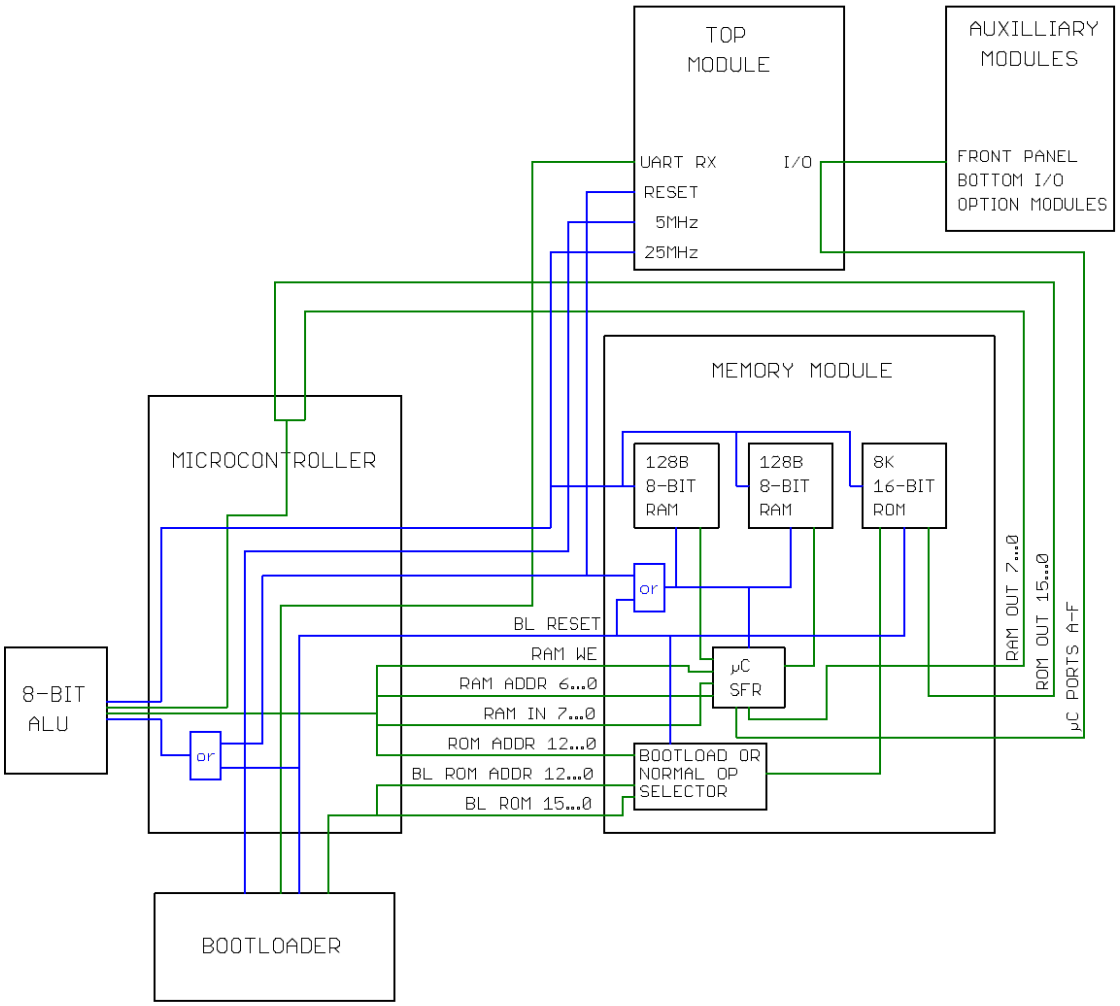
Module nine is the bottom I/O module which includes the digital I/O and the two RS422 serial ports. First, this module handles the I/O signals, which are usually coming from the EtherCAT IP-core through the µC-interface and then connected to each I/O buffer along with the diagnostic signals (if the buffer is configured as an output). Second, the module has its own sub-modules for handling the serial ports. The sub-modules include a baud rate generator together with one transmit and one receive unit. These sub-modules are not implemented in the first version of the product but will be added later on if needed.

Module 5 and 6

The fifth module in the picture is the memory module of the system. This module contains instantiations of the two microcontroller RAM-banks as well as the program ROM. Along with these; this module also connects the external EEPROMs of the option boards to the microcontroller RAM as well as handles all the microcontroller

I/O-ports and SFRs. The two latter are also both connected to specific RAM-addresses. Both the RAMs and ROM are synchronous and edge sensitive to the 25MHZ clock fed in to the module.

The sixth module of the system is the microcontroller module, which itself contains an ALU and a boot loader sub-module. The microcontroller in this project is a stripped and modified VHDL-implemented copy of the old PIC16F84 microcontroller [20]. Explained here is at first hand the microcontroller interfaces to other modules and not the implementation itself. Also, some details how the microcontroller and the boot loader are used will be discussed more thoroughly in chapter 4.2.1.. The microcontroller module is the top-module in this sub-system containing no logic itself and therefore works only as a route-through module connecting the memory to the ALU-module and to the boot loader-module. The ALU module executes the instructions it reads from the program ROM-memory and accordingly communicates with the RAM-memory. The boot loader module has an UART interface which is used to erase and write to the program memory if a software update is uploaded. The UART interface can be expanded in the future if necessary to read from the EtherCAT module or any other appropriate data interface. The following picture will give a clearer view of the memory block, the microcontroller and the boot loader and their interconnections and functions.



Picture 11. Signal connections between some of the VHDL modules

For clarity, in picture 11 the green lines indicate single- or bus-signal connections between the modules and the blue lines are clock or reset signals. Shown in the picture are all the memory interface signals from and to the microcontroller, along with some single signals of the system, such as the clock, reset and UART signals. Also shown are the internal functions of the memory which include the special function register of the microcontroller (SFR) and the boot-load-manager which erases and writes to the ROM. The SFR handles the RAM-bank selecting along with some but not all of the functions included in the PIC16F84-type microcontroller's SFRs [p. 7-12, 20]. Some of the functions that are not needed in the design are intentionally left out in order to reduce code-size and complexity. However, the choices made concerning what to implement and what not to implement will not be discussed.

As can further on be seen in the picture, there are two OR:ed reset signals in the system, one external and one coming from the boot loader. The function of the external reset is obvious but the boot loader reset is only active when the UART-unit has received a specific reset command. Finally, the µC ports are connected internally to the SFR part of the RAM in order to give them physical RAM-addresses which can be accessed by the microcontroller program. From there, the port signals goes through the top-module to their specific destinations, which can be the front panel, option modules and bottom I/O, for example.

Module 7

Module 7 stands for the module dealing with the four individual option module connectors. Each individual sub-module corresponding to a specific option module connector has to contain all possible logic needed to communicate with all the different modules that can be connected. To implement this, each option module is equipped with a small EEPROM which stores information telling the motherboard what kind of module is connected. Each individual module's EEPROM is read at startup by the microcontroller and according to the module ID appropriate logic is assigned to handle the option boards' I/O pins for communication. This function makes it possible to connect any option module to any of the four places and also allows one to have several option modules of the same type connected at the same time. It also makes it possible to add or remove option boards later on to the product without any firmware updates because all the needed logic is already inside the FPGA. The above information about logic association is true for option modules containing high-speed functions, which is usually the case with for example encoder reading. However, to save some logic, modules not requiring high-speed data transfer could be handled by the microcontroller I/O ports. In such a case a tradeoff is made between the microcontroller's RAM/ROM-memory requirements and the used amount of logic in the FPGA. Finally, the option module data is not only connected to the EtherCAT µC-interface but also connected to the memory module which makes it possible to read / write data through the front panel buttons + LCD-interface.

Module 8

Module 8 is the module driving the front panel of the product. The front panel itself consists of a PCB populated with an LCD display, a few buttons and a number of LEDs. These are all driven and read from the motherboard in a serial fashion. This means that a specific bit-sequence accompanied with a clock signal is fed through some of the pins of the front panel connector. Each bit in the sequence corresponds to a specific LED row/column, LCD pin or button. As a result, it is up to the logic of the front panel module to assemble the bits in the right order in the sequence while continuously

driving the sequence to the front panel at a decent refresh rate. The input data from the buttons is written on the fly onto the sequence as it passes through the panel logic and returns to the motherboard. The function can be compared to a shift-register. The bits to be included in the sequence are fed to this front panel module from a few of the microcontroller's ports.

4.1.4 Working with Xilinx FPGA design tools

For the basic FPGA designer, the Xilinx (or any other Vendor's) tools usually work fine without the need of interference from the user. In short, this means that the only thing the user has to do is to write the code and assign the physical I/O-pins of the FPGA package and leave the code synthesizing, translation and the FPGA placing and routing to the automatized software. This is often the case when the FPGA design easily meets all internal and external timing constraints and all the logic fits well inside the FPGA i.e. the device utilization is low to moderate. If this is not the case, there are user configurable options in each step of the process from VHDL code to FPGA programming file. With these options the user can choose between things such as the optimization grade of the code, the effort level of the placer & router and other different options to improve for example timing. The tradeoff with these additional options is often that the synthetizing time grows significantly. If these configurations do not help, a last resort including more radical measures can be done, which include for example placing the logic onto the FPGA manually and performing different timing analyzes to help solve the problem.

However, in this project at the time of this writing there was no need to interfere with the default options, as the FPGA utilization grade is quite low < 50% at the time. As a remainder, the FPGA chosen at this prototype stage was the largest possible in the series. However, because of the low utilization it could be possible to choose a smaller one for the production versions as the software develops. This choice could possibly then give raise to the problems described above in the last paragraph. However, it is beyond the scope of this thesis to discuss the possible counter-measures etc. if such a situation would arise.

4.1.5 Future expansion

Some of the possible future improvements concerning the programming were discussed in the earlier chapter, such as replacing the µC-interface with a microcontroller and communicating with the memories on-board. However, some even more radical changes to the overall VHDL system can be made. One of the most interesting things that could give more value to the product at the time is that the FPGA could possibly contain a more efficient processor and together with it implement the intelligent part of a complete servo drive. If this could be put off, the module could possible behave as one or more servo drives and use very simple motor drives as the power output stages, resulting in a cheap servo system providing some of the same modification possibilities as with the numeral option modules and customer specific software. However, this will not be further discussed in this thesis and is only mentioned here as an example for the many possibilities the VHDL software gives rise to.

4.2 Other additional software

In contrary to the VHDL coding, there are some other parts of the project that needs software in order to work. One of them, the microcontroller, was discussed in the earlier chapter. The microcontroller naturally needs a machine code program, which in this project was written in a combination of C and Assembly and then assembled by a tool capable of assembling C code for microcontrollers [21]. The other software is for the EtherCAT master, which needs a markup-file written in XML in order to work together with our slave. This XML-file can shortly be described as a description file for the master telling what kind of a slave is connected and what kind of data transfer is used and what other properties it has. Coding and building EtherCAT projects on the master side in software such as CodeSYS or similar will not be covered in this thesis.

4.2.1 Microcontroller C and Assembly

The microcontroller, which handles a good part of the internal functions of the module, will be further on discussed in this sub-section. The microcontroller was chosen to be developed and implemented in this project because it saves a lot of logic in the FPGA by moving a lot of the functionality to a processor program instead of doing the same things with plain logic. It also significantly eases the process of implementing the functions and modifying them later on. As already known, the microcontroller is a striped and modified VHDL implemented PIC16F84, using the 14-bit length op-code instruction set typical for the PIC16Fxxx family. In practice, this means that any type of microcontroller of the 16F family could be implemented easily by making relatively small modifications to the VHDL code. As could be seen in picture 11 earlier, the program ROM is 16-bits wide which leaves two empty bits in the op-code. These could be used to implement own instructions, which could come in handy if the program has to do a specific function which is easier or more efficiently implemented directly in VHDL.

The microcontroller is mainly programmed in C and the code assembled by the compiler given in [21]. However, some parts have to be implemented in Assembly directly because the VHDL microcontroller is modified from the original one and the C compiler naturally isn't designed to handle such cases. These modified parts are the own VHDL implemented instruction codes and some of the port communication parts, as the modified microcontroller has more I/O ports than the original one. Assembly can also be used to implement timing critical parts, which usually are harder to write in plain C-code. A good thing about the compiler mentioned earlier is that it can handle mixed code written in both Assembly and C inside the same code file. The specific microcontroller program used inside the product will not be discussed in the thesis.

As a reference for 8-bit microcontroller program writing in C, the book found in [22] was used. This book covers the parts of Ansi C which are applicable to the microcontrollers in question (16F series). Further on the book gives some advice using the CCSC compiler, which however not will be covered here.

4.2.2 XML markup file for an EtherCAT master

An XML markup file used by the master in an EtherCAT system is needed to provide information to the master about the slave(s) connected to the system. The ESI i.e. EtherCAT slave information included in the XML-files ranges from technical information such as number of sync-managers, PDO-mapping and other features to more informative info such as Vendor IDs, product number and bitmaps containing Vendor logos etc. The build-up of the XML-files follows a schema specific for EtherCAT. From the information provided by the slave specific XMLs the EtherCAT master builds up the EtherCAT network information (ENI) of the system. More about the XMLs and their structure can be found on the EtherCAT group homepage in [4], but it requires membership to access to the files.

The ESI for the motherboard was designed to consist of two different XML-files because of its modular build-up. The first XML-file contains information such as the Vendor, product number, version, Sync-managers and PDO-mapping for the bottom I/O and option modules. The additional file contains more specific information about every option module that can be connected to any of the four option module connectors. This gives us an easier way to include the slave into the master software by first adding the base module (motherboard) and then further on appending the different option modules used in the specific system.

4.3 Testing and evaluation of software

To begin with, testing the VHDL software can roughly be divided into two main methods, both with their pros and cons. The first one is simulating the VHDL software with a simulator such as ModelSIM [23] and observe the results by adding test-signals to the module under test. The other method is to directly download the code to the target FPGA and observe the operation together with an oscilloscope and / or other indicators. The first method can be faster in some cases and allows the user to see the behavior of any arbitrary signal in the system easily on a PC, which eases bug-hunting significantly. However, there are some drawbacks compared to downloading the code to the FPGA and testing the software in real hardware. First off, as a result of the internal placing and routing, there will be some hardware delays inside the FPGA which can affect the most timing critical systems. These effects can hardly be predicted without synthetizing the VHDL software to the target FPGA and reading the timing reports. Another disadvantage, related to this project, is that the EtherCAT IP-core can't be simulated because the source code is encrypted.

Testing the complete VHDL program which was used in this product was done in many stages. Due to the modular build-up of the VHDL code it was possible to test every module separately and when confirmed working, the module could be added to the system. Because of the relatively easy structure of all the modules except the microcontroller and EtherCAT IP-core, every module except these were tested in hardware directly. However, the microcontroller was first tested with the ModelSIM simulator to ensure proper operation because it contains many complex internal signals which would have been hard and effortful to trace using an oscilloscope. Every time when a new module or sub-module was tested and confirmed to be working, it was added to the complete VHDL structure and synthetized together with all the other modules. This was done for reasons described before, namely that the internal placing and routing can affect the results. As a conclusion, it will only be stated here that all the

different VHDL modules was tested one at a time and they were confirmed to be working together as excepted. No more details of the tests will be given, because they are rather a matter of writing the initial code, testing, bug fixing, testing again and so on.

The microcontroller C-code was first of all compiled by the CCSC compiler [21] described before and then downloaded through a PCs serial port to the FPGA internal memory. Testing was done completely in hardware by observing the function and using an oscilloscope for signals not otherwise detectable.

The XML-file for the master was harder to test because there is no compiler-type software which tells what possibly went wrong other than syntax errors. Due to this fact the XML-file was made by trial and error together with the Beckhoff EtherCAT master software TwinCAT [24].

5 Results and conclusions

The purpose of this work was to give insight into the key parts of the design of a new product to be included into an existing product family managed and developed by the company SKS Control Oy. The new module to be designed was intended to replace a number of other older modules of the product family, while simultaneously adding new properties and features, such as using the EtherCAT protocol for communication. The work included everything from the theory behind the product features to the key parts of the practical product development, including design phases such as designing the electrical schematics, PCB and software and finally testing the near production ready product. By near production ready it is meant that the prototype discussed in this work differs from the final production version by only less significant minor details.

5.1 Testing and confirmation of function

At the time this is written, the product was tested as described in the hardware and software sections of this thesis, but some additional testing is yet to be done. To begin with, the product has to be tested in different environments similar to the real operational conditions, such as inside an electrical cabinet together with servo drives which produce some unwanted noise, for example. Secondly, the product has yet to be tested for EMC compliance in order to fulfill the requirements for the CE-mark [25]. Additional to this, it is possible but not necessary to test the product by a certified EtherCAT test center in order to get an EtherCAT conformance certificate, which allows the vendor to sell the product as an EtherCAT conformance tested product. Although, it is still an open question if the conformance certificate is considered relevant in this project.

5.2 Future work

In contrary to the future testing discussed in the last section, some additional hardware and software work are also to be done. As already known, the product is modular in structure which allows for the possibility to develop any number of option modules that can be connected to the motherboard. This among others things is one of the things to which effort will be put before and after the product is released to market. The FPGA along with the microcontroller it contains makes it possible to change the hardware description and the software for the microcontroller whenever needed. This means that a lot of the future work will involve these soft wares in order to make a product that could easily be updated to serve the needs of different customers. In other words this means customer specific software.

5.3 Final summary

The main objective of this work was to discuss the development of a modular EtherCAT field-bus module. The work begun with a theoretical overview of the technology involved in order to gain enough understanding to get started. The technology behind the product was tested in an earlier stage with breadboards and evaluation kits. It was seen from these earlier test results of the hardware design that adapting the same technologies together into a single motherboard PCB was successful. After this, the thesis focused on the software design part, were §an initial FPGA and microcontroller software were discussed. The features required by the initial hardware description software and the microcontroller software were seen to be successfully implemented to the motherboard's FPGA. As a result of all this, it can be said that the near production-ready product fulfilled all the requirements set in the beginning with very little obstacles throughout the design process.

References

[1] Beckhoff, EtherCAT technology group established. PC-control – the new automation technology magazine, web magazine, 2004, April, issue 1. Cited 9.10.2012. URL: http://www.pc-control.net/pdf/012004/pcc_0104_e.pdf.

[2] Beckhoff, Ethernet puts on the pressure. PC-control – the new automation technology magazine, web magazine, 2004, April, issue 1. Cited 9.10.2012.URL: http://www.pc-control.net/pdf/012004/pcc_0104_e.pdf.

[3] The EtherCAT technology group, EtherCAT brochure. Web document. Updated May 2012. Cited 9.10.2012. URL: http://www.ethercat.org/pdf/english/ETG_Brochure_EN.pdf

[4] Homepage of the EtherCAT technology group. Cited 9.10.2012. URL: http://www.ethercat.org/en/tech_group.html

[5] Homepage of Beckhoff, Hardware Data Sheet - Slave controller IP core for Xilinx FPGAs, Web document. Cited 9.10.2012. URL: http://download.beckhoff.com/download/Document/EtherCAT/Development_products/EtherCAT_IPCore_Xilinx_V2_04a_Datasheet_all_v1i0.pdf

[6] The EtherCAT technology group, EtherCAT – the Ethernet fieldbus. Web document. Cited 9.10.2012. URL: http://www.ethercat.org/pdf/ethercat_e.pdf

[7] The EtherCAT technology group, EtherCAT – the Ethernet fieldbus, Implementation aspects. Web document. Cited 9.10.2012. URL: http://www.ethercat.org/en/technology.html#5

[8] Clive Maxfield, Newnes. 2008. FPGAs – Instant access. Linacre House, Jordan Hill, Oxford OX2 8DP, UK. Elsevier. Pages 204. ISBN-13: 978-0-7506-8974-8.

[9] Allegro Microsystems Inc, A4403 Datasheet. Web document. Cited 6.1.2013. URL: http://www.allegromicro.com/~/Media/Files/Datasheets/A4403-Datasheet.ashx

[10] Infineon Technologies AG, ISP752R Datasheet. Web document. Updated September 2008. Cited 7.1.2013. URL: http://www.infineon.com/dgdl/ISP752R_DS_11.pdf?folderId=db3a304412b407950112b408e8c90004&fileId=db3a304412b407950112b4295d1040cb&sId=db3a30433bbd3b5f013c15fdd2150403

[11] Würth elektronik, WE RJ45 LAN transformer prod.num. 74980111211 datasheet. Web document. Updated September 2011. Cited 11.1.2013. URL: http://katalog.we-online.de/pbs/datasheet/74980111211.pdf

[12] Homepage of Xilinx, Spartan 3E FPGA family datasheet. Web document. Updated October 2012. Cited 14.1.2013. URL: http://www.xilinx.com/support/documentation/data_sheets/ds312.pdf

[13] Homepage of CadSoft Inc. Web document. Cited 15.1.2013. URL: http://www.cadsoftusa.com/eagle-pcb-design-software/

[14] Homepage of Elprintta Oy. Web document. Cited 15.1.2013. URL: http://www.elprintta.fi

[15] Mitzner, Kraig. 2009. Complete PCB design using OrCAD Capture and PCB editor. Oxford, United Kingdom: Elsevier Inc. 471 p. ISBN 978-0-7506-8971-7.

[16] Homepage of Prinel Oy, Design tips. Web Document. Cited 22.1.2013. URL : http://www.prinel.fi/suunnitteluohjeet

[17] Homepage of Beckhoff, Application note – PHY selection guide. Web document. Updated September 2012. Cited14.1.2013. URL: http://download.beckhoff.com/download/Document/EtherCAT/Development_products/AN_PHY_Selection_GuideV2.1.pdf

[18] Micrel Inc, KSZ8001L Datasheet. Web document. Updated March 2006. Cited 11.1.2013. URL: http://www1.futureelectronics.com/doc/MICREL%20SEMICONDUCTOR/KSZ8001L.pdf

[19] Ashenden, Peter J. 2008. The designer's guide to VHDL. Third edition. Morgan Kaufmann publishers, USA: Elsevier Inc. 909 p. ISBN 978-0-12-088785-9.

[20] Homepage of Microchip Technology Inc, PIC16F84A datasheet. Web document. Updated 2001. Cited 27.2.2013. URL: http://ww1.microchip.com/downloads/en/devicedoc/35007b.pdf

[21] Homepage of Custom Computer Services, Inc. Web document. Updated 2013. Cited 28.2.2013. URL: http://www.ccsinfo.com/

[22] Martin P. Bates. 2008. Programming 8-bit PIC microcontrollers in C with interactive hardware simulation. Newnes, Burlington MA, USA: Elsevier Ltd. 278 p. ISBN 978-0-7506-8960-1.

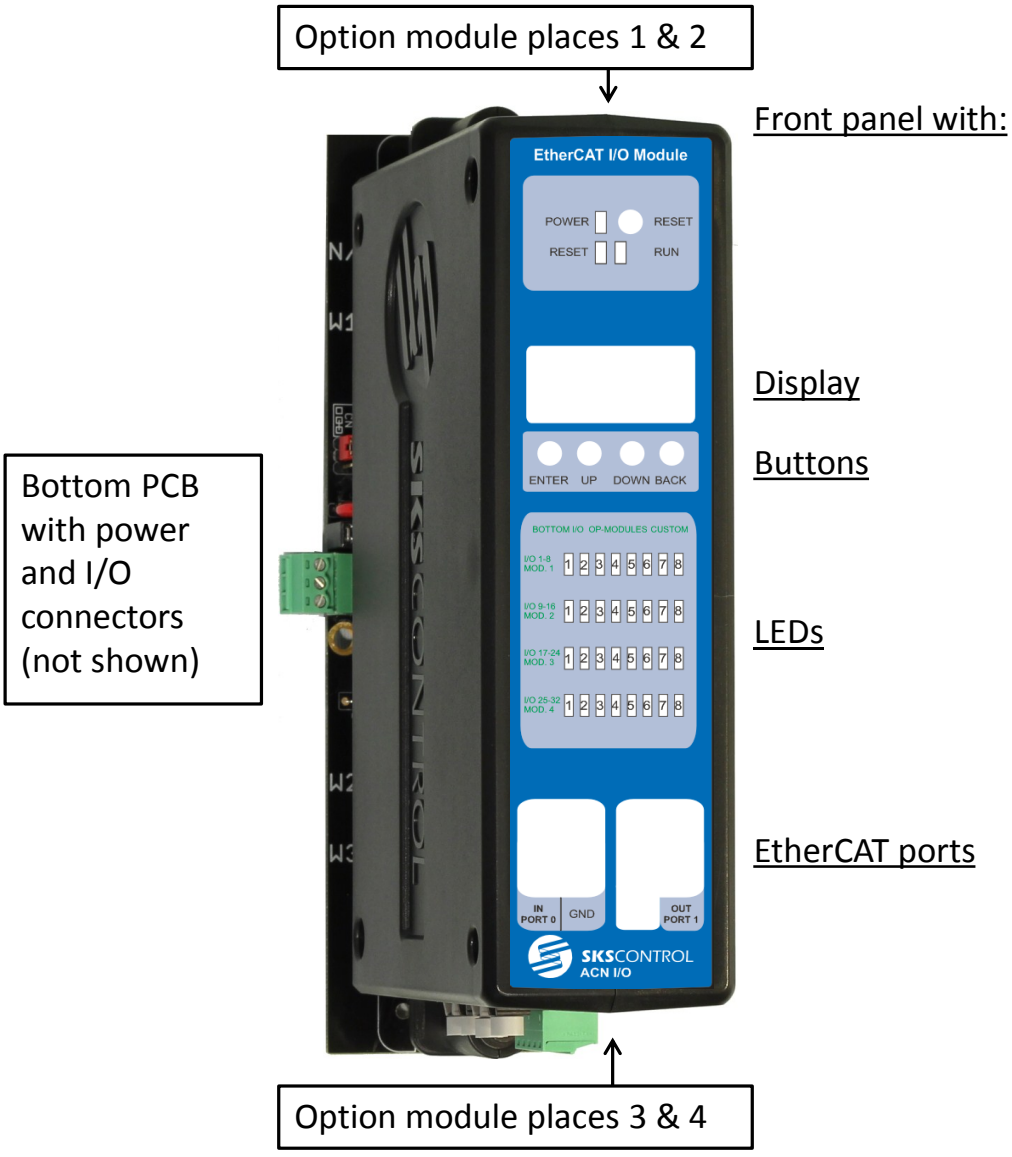
[23] Homepage of the Menthor Graphics ModelSim software. Web document. Updated 2013. Cited 4.3.2013. URL: http://model.com/

[24] Homepage of BeckHoff, TwinCAT 3 software. Web document. Updated 2013. Cited 4.3.2013. URL: http://www.beckhoff.com/english.asp?twincat/twincat-3.htm

[25] Homepage of the European commission, CE marking. Web document. Updated 2013. Cited 5.3.2013. URL: http://ec.europa.eu/enterprise/policies/single-market-goods/cemarking/index_en.htm

Appendix

A.



The complete product inside the plastic enclosure together with the surrounding PCB boards and a preliminary layout of the front panel.